# CA-IDMS®

# Features Guide

## 15.0

Computer Associates™

# Contents

## Chapter 1: Introducing Release 15.0



## Chapter 2: Upgrading to Release 15.0

# Chapter 3: Exploiting the Parallel Sysplex Architecture

# Chapter 4: Other Release 15.0 Enhancements

# Appendix A: New and Revised DCMT Commands

# Appendix B: New and Revised User Exits

## Appendix C: CICSOPT Macro



## Appendix D: New and Revised SQL Statements

# Appendix E: Revised Dictionary Records



# Appendix F: Defining and Using Procedures

## Appendix G: Sample COBOL Procedure



## Appendix H: Revised System Tables

Chapter 1

# Introducing Release 15.0

## Welcome

Welcome to CA-IDMS Release 15.0. This release incorporates many new features to enhance your use of CA-IDMS, including:

- Parallel sysplex exploitation
- Enhanced non-stop processing
- Improved performance and productivity
- SQL enhancements

This chapter includes a brief overview of each of the 15.0 features and provides a high-level explanation of the upgrade requirements. The remaining parts of this guide describe the features in detail:

| Part | Content |
|---|---|
| Chapter 2, Upgrading to Release 15.0 | Describes actions and considerations related to upgrading to Release 15.0. |
| Chapter 3, Exploiting the Parallel Sysplex Environment | Explains Data Sharing Groups, shared update access to data, shared queues and enqueued resources, dynamic routing enhancements, and related considerations. |
| Chapter 4, Other Release 15.0 Enhancements | Describes the non-stop processing enhancements, usability and performance enhancements. |
| Appendix A, New and Revised DCMT Commands | Provides full syntactical descriptions of DCMT commands that are new or revised for Release 15.0. |
| Appendix B, New and Revised User Exits | Provides full descriptions of User Exits that are new for Release 15.0. |
| Appendix C, CICSOPT Macro | Documents the CICSOPT macro that is new with Release 15.0. |

| Part | Content |
|---|---|
| Appendix D, New and Revised SQL Statements | Provides full syntactical descriptions of SQL statements that are new for Release 15.0. |
| Appendix E, Revised Dictionary Records | Provides a list of all new fields added to the dictionary records in support of Release 15.0 features. |
| Appendix F, Defining and Using Procedures | Explains how to define and invoke procedures. |
| Appendix G, Sample COBOL Procedure | Provides a sample procedure definition, a sample procedure written in COBOL, and a sample procedure invocation. |
| Appendix H, Revised System Tables | Identifies the changes made to the system tables in support of Release 15.0 features. |

# Parallel Sysplex Exploitation

CA-IDMS 15.0 builds on the sysplex exploitation and open architecture enhancements made in previous releases of IDMS. Through exploitation of Coupling Facility structures in a parallel sysplex environment, CA-IDMS Central Versions are now able to share update access to the same database areas.

A Data Sharing Group (DSG) is a new concept introduced with IDMS 15.0. There is no syntax required to define a Data Sharing Group. Rather a Data Sharing Group comes into existence when two or more CVs are started with identical values for the new SYSIDMS parm of DSGROUP=*group-name*. Members of a Data Sharing Group enjoy many privileges including the ability to share update access to areas, share access to the queue area, enqueue/dequeue global resources across systems, and broadcast messages and DCMT/DCUF commands to other members of the DSG.

Members of a Data Sharing Group can share update access to an area provided that:

- Data Sharing is enabled for the area
- Associated files for the area are assigned to shared cache
- Attributes for the area and associated files are identical in all systems

Internally, each participating CV must now perform global transaction and page locking as well as global deadlock detection and resolution for shared areas whenever a contention occurs. This is managed through the use of Coupling Facility structures. A cache structure must be defined for the buffers for the shared files and a list and a lock structure must be defined for use with global locking. XCF group services and signaling services are used by CA-IDMS for sending and receiving messages and monitoring Data Sharing Member status.

If one member of a data sharing group abnormally terminates, any records that it had updated but not committed prior to the failure, remain locked from access by other data sharing members until the CV has been warmstarted. It is important, therefore, to restart the failing CV as soon as possible to minimize its impact on other members of the group. A new Merge Archive utility is provided to merge the journal files of all data sharing members prior to beginning a roll forward recovery operation. With these exceptions, journalling and recovery are the same regardless of whether or not data sharing is being used.

# Enhanced Non-Stop Processing

Many features are introduced in IDMS 15.0 for enhanced non-stop processing including:

- The ability to dynamically add or increase the size of a storage pool or program pool
- The ability to dynamically specify the external wait time for a task
- The ability to modify the queue depth field used to tune multitasking performance
- The use of expanded system counters for number of system tasks, times at max tasks, and times SOS
- Toleration of database I/O errors during warmstart
- The automatic restart of RHDCDEAD (the deadlock detector) if it abends

# Improved Performance and Productivity

For improved performance and productivity the following features are provided:

- The use of XTIOT for an increased number of datasets that may be allocated to one CA-IDMS job
- Control over the use of dynamic allocation in local mode
- Printer formfeed options specifiable in sysgen at the system level as well as at the LTERM level

- The ability to run Unload, Reload, and Fastload with duplicate record ids defined in the same subschema
- A runtime message at the time of a stall which includes the program name of who is holding a dbkey resource
- A DCMT command to vary an area permanently to retain its new status
- Several vary area enhancements including the ability to cancel an outstanding vary operation and to force a vary operation to occur immediately
- The ability to quiesce one or more areas
- Easier installation and maintenance procedures for the CICS interface
- Enhancements to the CICS interface to support dynamic routing between CICS and IDMS for both SQL and non-SQL sessions
- An enhancement to the CA-Endevor/DB archive and compress facility to support modification of dictionary identification information in Confirmation Change Lot entries
- Management of User Attribute Blocks (UABs) for improved storage and CPU utilization
- The collection and display of data sharing statistics by the CA-IDMS Performance Monitor Interval Monitor
- CA-ADS language extensions that enable an application to handle assignment and computation errors
- A new CA-OLQ user exit that enables users to review and modify navigational DML commands prior to their execution

# SQL Enhancements

For open architecture, SQL enhancements include:

- SQL DDL syntax for CREATE PROCEDURE as well as EXEC SQL CALL PROCEDURE syntax for the runtime invocation of programs written in COBOL, PL/1, or Assembler
- Support for columns to be assigned the result of a subquery on an SQL UPDATE statement

# CA-IDMS 15.0 Upgrade Requirements

Upgrade requirements for CA-IDMS 15.0 are minimal for clients who are not exploiting Parallel Sysplex features. Installation of a new SVC is required. Installation procedures for the CICS interface have been streamlined to be completely under the control of SMP/E. (Clients who have made modifications to this interface will be able to continue to modify IDMSINTC outside of SMP/E.) The Convert Catalog utility must be run to update the catalog in place to pick up new definitions. DDLDML dictionary migration is not required; however, IDMSDIRL should be run to pick up the new definitions for fields in the dictionary that were previously defined as FILLER fields.

For clients using the Shared Cache or Dynamic Run Unit routing features (first available in CA-IDMS 14.0), the Coupling Facility structures used by these features have changed for improved performance. Systems using these features and sharing the same coupling facility structures must therefore all be at the same release level, i.e., 14.x or 15.0.

For improved debugging, the default for system snaps is now an SVC dump. Prior to release 15.0, an SVC dump would be created only if optional apar bit OPT00176 were set. The meaning of this bit is now reversed. In IDMS 15.0, setting OPT00176 will disable the SVC dump and direct the system snap to the log.

The journal file format has changed for IDMS 15.0. Journals must be reformatted for IDMS 15.0. Journals created by IDMS 15.0 are not useable by pre-IDMS 15.0 systems and vice versa.

The master suspend queue used by the CA-IDMS Dictionary Module Editor (DME) product has been renamed from “DME-SUSPEND-QUE” to “ _DME-SUSPEND-QUE”. All suspended DME sessions should be saved to the dictionary before upgrading to CA-IDMS 15.0.

To use the data sharing feature, a close examination should be made of your applications to determine their readiness for the use of shared resources. If the queue area is declared as sharable, then all queues (with the exception of some internal system queues) will be accessible to all other CVs in the Data Sharing Group with a shared queue area. Also, all resources for which ENQUEUE and DEQUEUE commands are issued by CVs in a Data Sharing Group are considered global. Queues and resources that cannot be shared must be included in an exception list.

Chapter 2

# Upgrading to Release 15.0

## Overview

This chapter describes the actions that must be taken and the considerations involved in upgrading to Release 15.0. You can upgrade to CA-IDMS Release 15.0 from CA-IDMS Release 10.0, 10.1, 10.21, 12.0, 14.0, or 14.1. The conversion utilities provided for CA-IDMS Releases 12.0, 14.0, and 14.1 are included on the CA-IDMS Release 15.0 installation tape.

Take the following actions to upgrade the software to Release 15.0:

- Install the software into a new SMP/E configuration
- Install the new SVC delivered with Release 15.0
- Recompile all user-written programs that reference CA-IDMS control blocks or the journal file record layouts
- Run IDMSDIRL against each dictionary that contains the IDMSNTWK schema definition
- Initialize the journal files using the Release 15.0 FORMAT utility before starting a system
- Offload the log file using a pre-Release 15.0 ARCHIVE LOG utility or initialize the log file using the Release 15.0 FORMAT utility before starting a system
- Update the CA-IDMS task and program definitions using the sysgen compiler and the source members provided on the installation tape
- Review the use of SVC dumps for system snaps to determine the need for optional APAR OPT00176
- Change existing FASTLOAD format programs to supply the name of the record being loaded
- SQL users should issue the CONVERT CATALOG statement against each of their SQL-enabled dictionaries
- CA-IDMS Dictionary Module Editor (DME) users should ensure that all suspended sessions are terminated before starting a system under Release 15.0

- CICS users should review the considerations outlined below to determine what additional actions, if any, need to be taken
- Users that are currently exploiting Parallel Sysplex architecture through the use of shared cache or dynamic database routing should review the considerations outlined below
- OS/390 users should review user-written assembler subroutines invoked directly from COBOL and PL/I programs
- Users who invoke the duplex journal exit (IDMSJNL2) must change their exit to handle a journal buffer allocated in 31-bit storage

## Installing the Software

Follow the instructions outlined in the cover letter delivered with the installation tape. Be sure to install the software into a new SMP/E configuration.

## Installing the SVC

A new SVC has been delivered with Release 15.0. It should be used for all Release 15.0 systems. Since it is downward compatible, it may also be used for pre-release 15.0 systems.

## Recompiling User-written Programs

Several control block formats have been changed in Release 15.0. In addition, the record layouts of archive journal files have changed. Although in most cases, the changes simply entail the addition of new fields, it is recommended that all programs referencing CA-IDMS control blocks, such as user-written exits, be recompiled using the Release 15.0 source library.

## Updating Dictionary Descriptions

Some new fields were added to records in the DDLDML area using existing filler space. Although no dictionary conversion is necessary, you should update the definition of these records in every dictionary that contains the IDMSNTWK schema description. To do this, use the IDMSDIRL utility. For instructions on executing this utility, refer to CA-IDMS Utilities.

# Formatting Journal Files

The format of journal records has changed in Release 15.0. Fields have been added to every record to identify the time that it was created and the system that created it. Additional changes have been made to certain record types. These changes necessitate that journal files be initialized using the Release 15.0 FORMAT utility statement prior to their use with a 15.0 system. At startup, the system will ensure that the journal files have the correct format.

If it is necessary to fall back to an earlier release of the software, the journal files must be re-initialized using the FORMAT utility statement for that release. Warmstart will fail if an earlier version of the software attempts to use Release 15.0 format journal files.

# Offloading the Log File

The format of statistics records written to the log has been changed in Release 15.0. In order to distinguish these from prior versions, the release identifier in these records contains the string "R150". The Release 15.0 ARCHIVE LOG utility will issue a warning if it encounters a log record with an earlier release identifier. In order to avoid these messages and to separate logs from the older releases, you should offload the log file using your pre-Release 15.0 ARCHIVE LOG utility prior to starting a Release 15.0 system. Alternatively, if you do not need to retain the log information, you can initialize the log file using the Release 15.0 FORMAT utility.

# Updating Task and Program Definitions

The definition of certain CA-supplied tasks and programs has been changed for Release 15.0. You should update the system definition using the sysgen compiler and source members provided on the installation tape. This can be accomplished most easily by performing an upgrade install and copying the task and program definitions from SYSTEM 99. For more information on the upgrade install process, refer to the *CA-IDMS Installation Guide*.

# Controlling System Snaps

By default in Release 15.0, system snaps will cause an SVC dump to be taken rather than being written to the DC log. This reduces the length of time that the system is single-threaded when an error occurs. To force system snaps to be written to the DC log as in prior releases, you must apply optional APAR OPT00176.

**Note:** In earlier releases, OPT00176 caused SVC dumps to be taken since the default was to write system snaps to the DC log.

# Changing FASTLOAD Format Programs

Because the FASTLOAD utility statement now supports loading records with duplicate record identifiers, all existing format programs must be changed to supply the record name as part of the record descriptor information. For more information, refer to "Unload/Reload/Fastload Extensions" in the Ease-of-Use Enhancements section of chapter 4.

# Updating SQL Catalog Definitions

SQL users must use the CONVERT CATALOG utility statement to update the definitions of system tables.

The converted definitions are compatible with Release 14.0 and 14.1 versions of the software, but are not compatible with earlier releases. Consequently, it is important to backup the catalog prior to conversion if upgrading from Release 12.0 or 12.01.

If it is necessary to fall back to a Release 14.0 or 14.1 version of the software, no special action needs to be taken regarding the catalog; however, if falling back to a Release 12.0 or 12.01 version, the catalog (and database areas containing tables that are created or altered using Release 15.0) must be restored.

## Release 15.0 Changes

When a catalog is converted, the definitions of the following tables are upgraded from their Release 14.1 definitions and new columns in associated rows are initialized appropriately:

- SYSTEM.DMCL
- SYSTEM.DMCLAREA
- SYSTEM.DMCLSEGMENT
- SYSTEM.TABLE

Changes introduced in earlier releases of the software are applied if they have not already been made. Refer to the Release 14.0 Features Guide for a description of these earlier changes.

## Executing the Catalog Conversion Utility

The catalog conversion utility is invoked under the Command Facility by entering the following statement:


After a successful execution, the Command Facility issues one of two informational messages to indicate the status of the conversion.

If a catalog conversion is performed, the message indicates the number of rows of each type that were changed.

If a catalog conversion was not required, a message indicating this fact is displayed.

# Saving Suspended DME Sessions

Users of the CA-IDMS Dictionary Module Editor (DME) must ensure that no sessions are suspended when first starting a Release 15.0 system. If any such sessions exist, they must be terminated by saving the changes to the dictionary.

The reason for this consideration is that the name of the DME master suspend queue has been changed from 'DME-SUSPEND-QUE' to '_DM-SUSPEND-QUE'. The format of the suspend queue has also changed. To ensure that no sessions are suspended, you may use the QUED task code to determine whether or not this queue exists. If it doesn't exist, there are no suspended DME sessions. If it is necessary to fall back to an earlier version of the software, you must again ensure that no suspended DME sessions exist.

# Using the New CICS Interface

In Release 15.0, the IDMSINTC CICS interface is installed under SMP/E and can be maintained using SMP/E. Although most users will be able to take advantage of this improved delivery method, some users may still need to tailor their own interface by making source changes or may need to use an earlier version of the interface. For more information on these considerations, refer to "Streamlined CICS Interface Installation" in Chapter 4, "Other Release 15.0 Enhancements."

If using the new CICS interface, the following changes must be made to the CICS PPT:

- Replace IDMSTIMP with RHDCTIMP
- Add IDMSBRBK

**Note:** No changes are necessary if only the IDMSINTL CICS interface is used.

# Exploiting Parallel Sysplex

In order to provide enhanced exploitation of the Parallel Sysplex architecture, certain changes were necessary in Release 15.0 that impact existing users of shared cache and dynamic database routing.

One consequence of these changes is that users cannot share cache structures nor route database sessions between Release 15.0 and 14.x systems.

## Shared Cache Changes

The following changes impact shared cache users:

- The ZIDMSCACHE0001 list structure is no longer used
- The AVAILABLE shared cache option is no longer supported
- The format of the cache structures have changed

For more information on each of these changes and their impact on existing users, refer to "Shared Cache Enhancements" in the Sysplex Exploitation Enhancements chapter.

### Dynamic Routing Changes

The following changes impact dynamic database routing users:

- The number and names of the list structures have changed
- It is no longer necessary to define backend systems in a frontend system's node table

For more information on each of these changes and their impact on existing users, refer to "Dynamic Routing Enhancements" in the Sysplex Exploitation Enhancements chapter.

## Reviewing Assembler Routines

In order to provide increased control in an OS/390 environment, all SVCs issued from COBOL and PL/I programs executing under the control of DC/UCF are now trapped through the use of SVC screening. If the SVC is not supported, the task will fail unless corrective action is taken. Assembler programs that are linked with COBOL or PL/I programs should be reviewed as part of the upgrade to Release 15.0.

For more information, refer to "Controlling SVCs from high-level languages" in Chapter 4 of this manual.

## Revising the Duplex Journal Exit

Journal buffers are now allocated from XA storage. This allows more buffers to be allocated without adversely impacting below-the-line storage requirements. However, this change impacts existing duplex journal exits (IDMSJNL2 exits). They must be changed to address this buffer in 31-bit mode.

Chapter 3

# Exploiting the Parallel Sysplex Architecture

## Overview

Release 15.0 continues the exploitation of IBM's parallel sysplex architecture that was introduced in Release 14.0. Through the use of coupling facility services the following new features are now available in Release 15.0:

- The ability for multiple CA-IDMS systems to update data concurrently
- The ability to share queues and enqueue common resources across CA-IDMS systems
- The ability to broadcast system tasks to multiple CA-IDMS systems

Each of these new features depends on the implementation of a data sharing group to which individual CA-IDMS systems join as members. A data sharing group is a construct that is new in release 15.0. It was introduced specifically to enable the implementation of the above features.

In addition to new ways to exploit the parallel sysplex architecture, the existing support for dynamic routing has been enhanced to make it easier to use and shared cache now requires fewer resources in the coupling facility.

This chapter first describes what a data sharing group is and how it is defined and then describes each of the new and enhanced sysplex exploitation features in more detail.

## Data Sharing Groups

A data sharing group is a named collection of CA-IDMS systems within a sysplex. Each CA-IDMS system that is associated with a data sharing group is referred to as a member of that group. The name of the member becomes both the CA-IDMS system name and the node name for that system. A system can be a member of only one data sharing group at a time.

Data sharing groups allow you to:

- Share update access to a database. See "Sharing Update Access to Data" for more information.
- Broadcast commands to all members of the group. See "Broadcasting Commands" for more information.
- Share queues and enqueue common resources. See "Sharing Queues and Enqueued Resources" for more information.
- Monitor and report on all members of a data sharing group. See "Monitoring a Data Sharing Group" for more information.

# Designing Data Sharing Groups

The most significant benefit to a data sharing group is the ability for multiple CA-IDMS systems to update the same data concurrently. Satisfying your data sharing needs should therefore be the most significant consideration in designing your data sharing groups.

## Types of Groups

There are three basic types of groups: a homogeneous group, a heterogeneous group, and a hybrid group.

A homogeneous group is one in which all members are essentially the same. They support the same applications, they access the same databases in the same way, and they have the same security definitions. Members of a homogeneous group likely share the same system definition, using the cloned system capability. Since every member has access to the same resources, it doesn't matter on which member a given transaction is executed.

In a heterogeneous group, every member is unique in terms of the applications that it supports. Although some databases may be shared between members, other databases are unique to a given member. Each member may have its own system and security definitions. They are members of the same group in order to share update access to data, but otherwise are distinct systems. A given transaction must be directed to a particular member in order to ensure that it has access to the resources that it needs.

A hybrid group is one in which some members are clones of one another while other members are not.

## Homogeneous Groups

A homogenous group can be thought of as a multi-part IDMS system. Facilities such as shared update access to data and shared queues enable applications to execute on any member of the group. The major benefits to this type of group are:

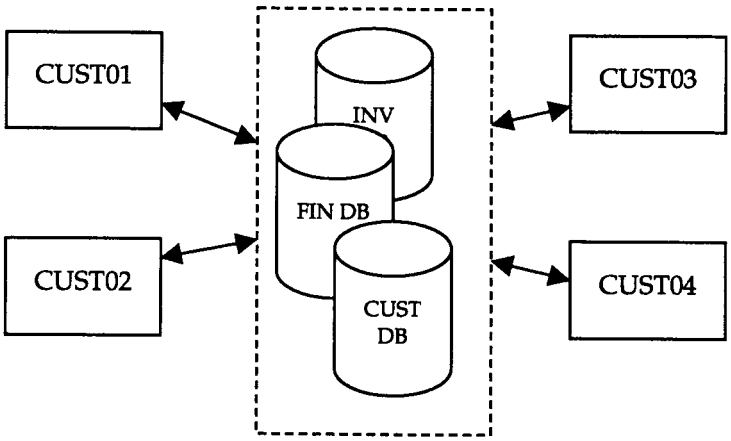
- The ability to adjust the number of members in response to changing workloads
- Fault tolerance in the event of a failure

If a group is reaching capacity in terms of transaction volumes, then an additional member can be started to handle some of the workload. Members might be added and removed on a periodic basis (daily, monthly, etc.) Or members may be permanently added as workload increases over time.

Homogeneous groups also provide fault tolerance in the event that a CA-IDMS system or an OS/390 image fails. Other members of the group (which may be executing on different OS/390 images) can continue to process applications, while recovery is taking place.

The degree to which applications update the same records within the database will determine the effectiveness of a homogeneous group. For example, if every transaction must update a one-of-a-kind control record, then there will be a high degree of contention for that record across members of the group. This will significantly increase the CPU overhead needed to process the transactions, since the members must communicate with one another in order to resolve the contention. Furthermore, if a CA-IDMS system fails while it holds an exclusive lock on the control record, then that record remains locked until the failing system has been recovered, thus preventing other transactions from accessing the record. If, on the other hand, transactions tend to access and update different records and pages in the database, with only the occasional overlap, then a homogeneous group should perform well and provide increased fault tolerance. In designing a homogeneous group, consider ways to segment the workload to minimize cross-member contention for resources.

The following diagram illustrates a homogeneous data sharing group. It consists of four members (CUST01, CUST02, CUST03, and CUST04), each of which share update access to the same set of databases (Inventory, Customer and Financial).



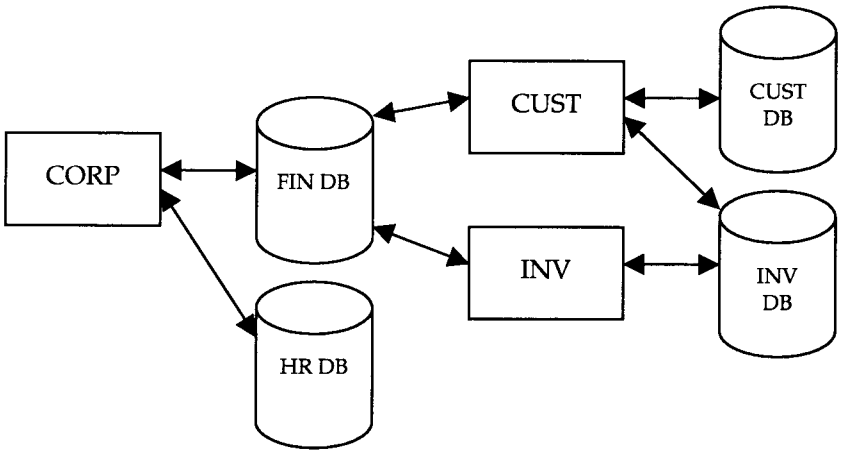

## Heterogeneous Groups

A heterogeneous group provides the ability to share update access to certain areas of the database from otherwise distinct CA-IDMS systems. It eliminates the need for other solutions such as application-level replication of updates or remote database access. It can alleviate the pressures of an increasing workload, by allowing it to be split along application boundaries even though some areas need to be commonly updated.

While contention for individual records and pages within shared areas may not be as likely in a heterogeneous group, the degree to which such contention occurs will affect performance. This should be one of the considerations in determining how to split your workload across members of a heterogeneous group.

Heterogeneous groups provide a degree of fault tolerance in that members that have not failed will continue to process transactions, some of which may update shared areas. Of course there is no fault tolerance for the portion of the workload that was being processed by the failed member.

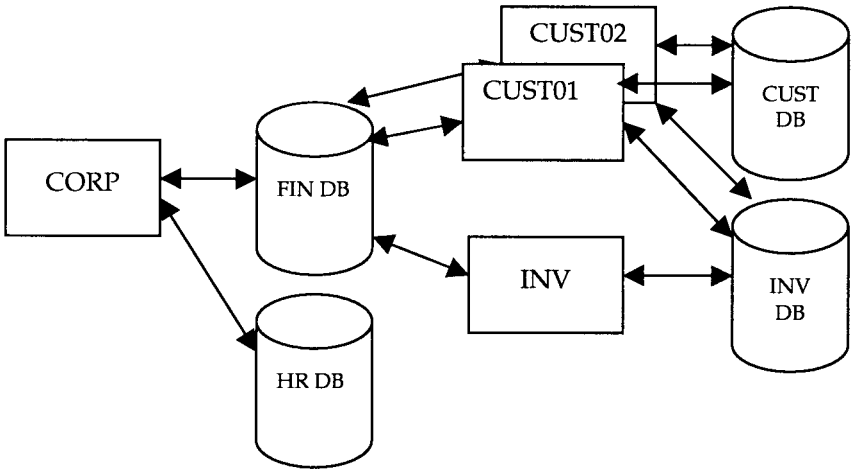
The following diagram illustrates a heterogeneous group in which every member accesses a distinct set of databases. Member CORP updates the Financial and HR databases. Member CUST updates the Customer, Financial, and Inventory databases. Member INV updates the Inventory and Financial databases.




## Hybrid Groups

A hybrid group can provide the benefits of both homogeneous groups and heterogeneous groups. It easily allows the addition of members to handle increased workload and provide fault tolerance for certain applications while allowing other members to continue processing different applications that have little contention for shared databases.

The following diagram illustrates a hybrid group that is similar to the heterogeneous group described above except that the CUST member is cloned so that two members CUST01 and CUST02 service the same set of transactions.




# Data Sharing Group Versus DBGroup

There is a difference between a data sharing group and a database or DBGroup. A data sharing group provides the ability to share update access to data. A DBGroup provides the ability to dynamically route database requests. At any one time, a CA-IDMS system can be a member of several DBGroups but only one data sharing group.

While it is possible that every member of a data sharing group also is a member of the same DBGroups, this is not a requirement and in fact, is unlikely unless the data sharing group is homogeneous. It would be more likely that members that are clones of one another would be members of the same set of DBGroups, since they can support the same types of transactions.

It is also possible for a DBGroup to span data sharing groups. CA-IDMS systems that are members of different data sharing groups or that are not members of any data sharing group can belong to the same DBGroup. While such a scenario may be unlikely, it highlights the fact that data sharing groups and DBGroups are technically unrelated.

For more information on dynamic routing and enhancements for Release 15.0, refer to Dynamic Routing Enhancements later in this chapter.

# Defining Data Sharing Groups

There is no explicit definition for a data sharing group; however, before a CA-IDMS system can become a member of a group, certain actions must be taken:

- The number of XCF groups that can be created in the coupling facility may need to be increased.
- The number of buffers used for XCF messaging may need to be increased.
- A list and lock structure must be defined to the coupling facility.
- Each CA-IDMS system that is to be a member of a group must be associated with that group through its startup JCL.
- Print task codes defined through the #UCFUFT macro must have a corresponding entry added to the LCLENQDQ module. This topic is discussed in Sharing Queues and Enqueued Resources later in this chapter.
- If members of the group are to share update access to data, changes must be made to the DMCLs of all group members. This topic is discussed in Sharing Update Access to Data later in this chapter.

## Selecting a Group Name

A data sharing group internally corresponds to an XCF group whose name is that of the data sharing group. The name of a data sharing group must be different from the name of any other XCF group within the sysplex.

Group names may be 1-8 characters in length and consist of characters A-Z, 0-9, $, # or @. Names that begin with SYS or UNDESIG are reserved and cannot be used. Names that begin with A-I may be used by the operating system and should be avoided. For more information refer to the appropriate IBM documentation.

## Configuring the Coupling Facility

Each data sharing group internally uses an XCF group and requires the definition of a list and a lock structure in the coupling facility. In addition, one or more cache structures must be defined if the data sharing group is to share update access to data or if shared buffering is to be used.

## XCF Group

XCF groups are not explicitly defined to the coupling facility, however the maximum number of XCF groups that can be active is part of CFRM policy. This maximum may need to be increased in order to implement a new data sharing group.

In addition, CA-IDMS uses XCF messaging to communicate between group members. It may be necessary to increase the number of buffers available for XCF messaging in order to accommodate the increased traffic. Refer to the appropriate IBM documentation for more information.

## CF Structures

The following information must be specified when defining a structure to the coupling facility:

- Structure name
- Structure size

### List Structure

The name of the list structure must be:

`CAIDMS`*`groupname`*`LI`

where *groupname* is the name of the data sharing group.

Use the following formula for estimating the size of the list structure:

```
Size = ROUND1M(SAREALS + SFILELS + QS*QUEUELS + QUIESLS + 256K)
```

where:

- SAREALS is the size of the shared area list. Compute its size as follows:
  - Compute for each shared area the area-element-size:

    ROUND256(184 + number-of-files-in-area * 35) +97
  - Sum all area-element-size's
- SFILELS is the size of the shared file list. Compute its size as follows:
  - Compute for each shared file the file-element-size:

    ROUND256(164 + number-of-areas-in-file * 6) + 97
  - Sum all file-element-size's
- QS = 1 if the queue area is shared; otherwise it is 0
- SQUEUELS is the size of the shared queue list. Compute its size as follows:
  - Number-of-shared-queues * 353

- QUIESLS is the size of the quiesce list. It is used when a DCMT QUIESCE command affects one or more shared areas. Its size is dependent on the number of parallel outstanding requests. The size of an outstanding request can be computed as follows:
  - ROUND256(92 + number-of-target-shared-areas * 6) + 97

**Note:** ROUND1M means rounding up to the next 1 mega-byte multiple; ROUND256 means rounding up to the next 256-byte multiple. E.g.,

ROUND256(198) = 256
ROUND256(256) = 256
ROUND256(258) = 512

The size of a list structure may be altered at any time using an operating system command. The ability to increase the size of the list structure is limited by the available space in the coupling facility. CA-IDMS does not allow rebuilding of the list structure.

### Lock Structure

The name of the lock structure must be:

`CAIDMS`*`groupname`*`LK`

where *groupname* is the name of the data sharing group.

The minimum size of the lock structure depends on the number of concurrent locks that will be placed on records in shared areas.

A coupling facility lock structure contains two types of information: a lock table and record data entries.

The lock table is used as a hash table for the purposes of detecting contention for a resource. Its size is determined by the LOCK ENTRIES and the MEMBERS parameters specified in the DMCL of data sharing group members. For more information on these parameters, see "Enabling Data Sharing" later in this chapter.

Record data entries are used to hold information about exclusive locks. Each exclusive global lock on a transaction resource has a corresponding record data entry. Applications which update many records before issuing a commit, will increase the requirement for record data entries. If the system runs short of record data entries, it will react by releasing proxy locks that are not in use. This will have a negative impact on performance. If, after taking this action, there are not enough record data entries available to satisfy a lock request, the issuing task will fail.

For more information on transaction locking, refer to "Lock Management" later in this chapter. For information on monitoring the available record data entries in the lock structure, refer to "Monitoring Data Sharing Groups" later in this chapter.

The size of a lock structure, must be large enough to hold both the lock table and the maximum number of record data entries that will exist at one time. The first CA-IDMS system that starts as a member of a data sharing group determines the size of the lock table. If the lock structure is too small to accommodate the lock table, startup will fail. Once the lock table has been allocated, its size remains the same until all group members have been shut down normally.

Whatever space remains in the lock structure after the lock table has been allocated is used for record data entries. The amount of space in the lock structure can be increased (or decreased), by using the SETXCF START ALTER command. The ALTER command only affects the amount of space available for record data entries. Task abends will result if there is not enough space to create a record data entry when one is needed. The amount of available space in the lock structure can be monitored by using the DCMT DISPLAY DATA SHARING command. For more information, refer to "Monitoring a Data Sharing Group" later in this chapter.

The minimum recommended lock structure size is 1,000k. This value may need to be significantly increased if areas are to be shared for update access. Use the following formula for estimating the size of the lock structure:

```
Size = LTE#*LTES + RD#*140 + 35K
```

where:

- *LTE#* is the number of lock table entries
- *LTES* is the lock table entry size as determined from the following table
- *RD#* is the maximum number of record data entries needed

| Maximum Number of Members in Data Sharing Group | Lock table entry size (*LTES*) |
|---|---|
| 7 | 2 |
| 8-23 | 4 |
| 24-55 | 8 |
| 56-119 | 16 |
| 120-247 | 32 |

CA-IDMS does not allow rebuilding of the lock structure.

### Cache Structures

At least one cache structure is needed in order to share update access to data or to enable shared buffering. Both of these capabilities require that each affected CA-IDMS file be associated with a cache structure. You may associate any number of files with a single cache or allocate a cache for each file.

The larger the cache structure, the more likely a page will remain in the cache, eliminating a disk access when the page is next needed by a system. Ideally, each file would be assigned to its own cache structure and that structure would be large enough to hold the entire file. However, this may not be practical due to the amount of space and the number of structures that would be needed. The more space that can be assigned, the fewer disk accesses that will be needed and hence the better the performance.

The size of a cache structure may be altered at any time using the SETXCF START ALTER command. The size may be increased, provided there is sufficient space in the coupling facility.

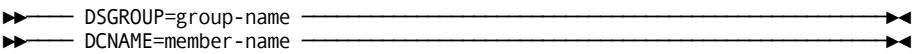
There are no additional requirements for the name of a cache structure beyond those imposed by the operating system.

CA-IDMS does not allow rebuilding of a cache structure.

## Specifying Group Membership

Each CA-IDMS system that is to be a member of a data sharing group must specify the name of the group and its membername in the SYSIDMS card image file in the startup JCL.

### Syntax




### Parameters

`DSGROUP=group-name`

Specifies the name of the data sharing group of which this system is a member. All CA-IDMS systems that are members of the same group must specify the same group name.

*group-name* must be a 1-8 character name consisting of characters A-Z, 0-9, $,# or @. Names that begin with SYS or UNDESIG are reserved and cannot be used. Names that begin with A-I may be in use by the operating system and should be avoided.

`DCNAME=member-name`

Specifies the member name of the system within a data sharing group. This name also becomes the system (node) name, overriding the value specified in the system definition.

*member-name* must be a 1-8 character name consisting of characters A-Z, 0-9, $,# or @.

### Usage

*Specifying a group name*: All CA-IDMS systems that specify the same group name are members of the same data sharing group and are therefore capable of sharing update access to data as well as exploiting other features associated with data sharing groups.

*Specifying a member name*: Member names must be unique within a data sharing group. Member names should also be unique across your environment since the member name becomes the system (node) name.

*Changing group and member names*: Once a CA-IDMS system has become a member of a data sharing group, it remains a member until the system has been shut down normally. This means that after an abnormal termination, the CA-IDMS system must be restarted with the same group and member name as at the time of failure. This also means that a failed group member cannot be restarted without a group and member name.

Similarly, a CA-IDMS system that was not a member of a data sharing group can become a member only if the system had previously terminated normally.

## Sharing Update Access to Data

The ability to share update access to data is referred to as *data sharing*. It allows multiple CA-IDMS systems to update specified areas of the database concurrently.

For data sharing to occur, each CA-IDMS system that shares update access must be a member of a data sharing group. (See "Data Sharing Groups" for more information) Only members of one data sharing group can have update access to a shared database area at one time. Other data sharing groups, other CA-IDMS systems that are not members of the group and local mode IDMS applications can access the area in retrieval mode only.

## Shared Area Requirements

In order for an area to be eligible for data sharing, the following attributes of the area and associated files must be identical in all sharing systems within a group:

- Page range, page group, and number of records per page
- Segment and area names
- Page size
- File mappings
- IDMS file names
- DSNAME and VOLSER of the associated disk files

Additionally:

- A shared area cannot be native VSAM.
- A shared area cannot be part of a dictionary controlled by CA-Endevor/DB.
- No two shared areas within a data sharing group can have overlapping page ranges within a page group.
- Within a data sharing group, the combination of DSNAME and VOLSER must be unique for all IDMS files associated with shared areas.

If these conditions are not satisfied, you must alter your DMCL and segment definitions before declaring the area to be shared. Failure to do so will mean that one or more members of the group will be unable to access the area.

These requirements are waived on any CA-IDMS system that is accessing the area in a transient retrieval mode regardless of whether or not the area has been designated for data sharing.

## Notify Locking Considerations

Notify locks are supported in a data sharing environment. If a transaction executing in one member places a notify lock on a record, it will be informed of any changes made to that record by other transactions regardless of where (in which member) the updating transaction executes. However, cross-member notification of retrieval is not supported. If an application relies on notification of retrieval, the database that it accesses should not be shared for update.

## Enabling Data Sharing

In addition to defining a data sharing group, the following tasks must be performed in order to enable data sharing:

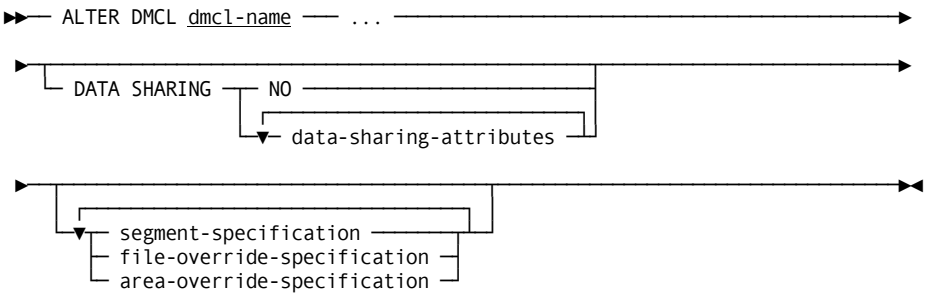
- The DMCL of every member of the group must be altered to indicate that data sharing is allowed
- Each area that is to be eligible for data sharing must be so designated. This can either be done in the DMCL or through DCMT commands.
- Each file associated with a shared area must be associated with a coupling facility cache structure. This can either be done in the DMCL or through DCMT commands.

## Altering the DMCL Definition

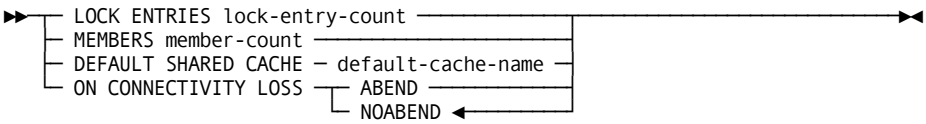
To enable data sharing, the DMCL definition for every system in the data sharing group must be altered to indicate that data sharing is allowed and to specify certain related information.

The DMCL definition may also be changed to indicate which areas are to be shared and the shared cache structure to be associated with their files. This information should be specified in the DMCL rather than through DCMT commands for areas that are always or typically shared among members of the group.

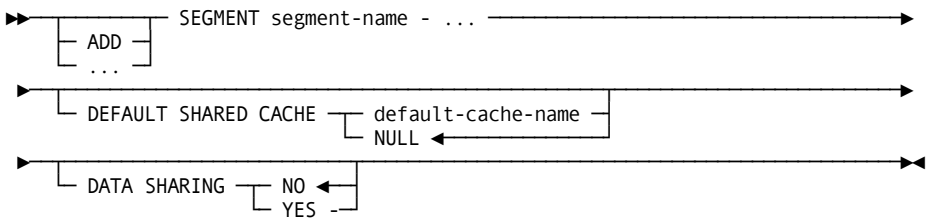
## Syntax



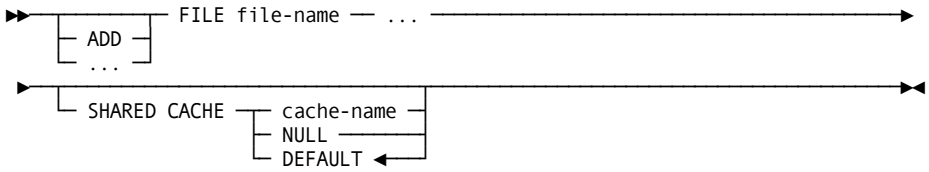

**Expansion of** data-sharing-attributes:



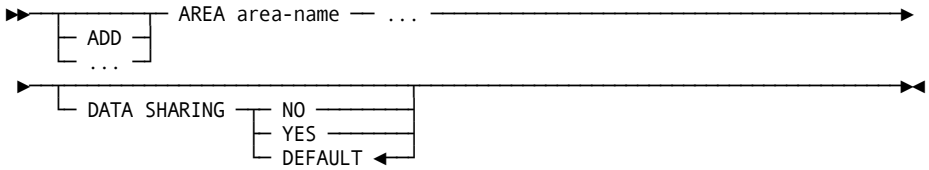
**Expansion of** segment-specification:




**Expansion of** file-override-specification:




**Expansion of** area-override-specification:




## Parameters

`DATA SHARING NO`

Removes data sharing information from the DMCL. If a system that uses this DMCL becomes a member of a data sharing group, the following default values will be assumed:

- lock-entry-count: 4096
- member-count: 7
- default-cache-name: null
- connectivity loss: NOABEND

`DATA SHARING` *`data-sharing-attributes`*

Specifies data sharing information for a CA-IDMS system using this DMCL.

If data sharing information is not specified, and the system becomes a member of a data sharing group, the default values shown above will be used for the individual attributes.

Data sharing information is ignored if a CA-IDMS system is not a member of a data sharing group.

LOCK ENTRIES *lock-entry-count*

Specifies the number of lock table entries that will be allocated within the coupling facility lock structure. The value specified must be in the range 4096 through 1,073,741,824. See "Specifying the number of lock table entries" below for more information on this parameter.

MEMBERS *member-count*

Specifies the maximum number of CA-IDMS systems that can be members of the data sharing group at one time. The value specified must be in the range 7 through 247. See "Specifying the maximum number of group members" below for the effect of this parameter.

DEFAULT SHARED CACHE *default-cache-name*

Specifies the default shared cache for any system using this DMCL. *Default-cache-name* must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.

The default shared cache for a system is used at runtime for any file associated with an area that is designated for data sharing, unless the file has an assigned cache.

This value has no affect on files that are not associated with a shared area.

ON CONNECTIVITY LOSS

Specifies what action the CA-IDMS system is to take when either a loss in connectivity to or a failure of the list or lock structure associated with the data sharing group is detected. ABEND — specifies that the CA-IDMS system is to abnormally terminate immediately. NOABEND — specifies that the CA-IDMS is to remain active in order to service non-datasharing related requests. NOABEND is the default if ON CONNECTIVITY LOSS is not specified.

segment-specification

DEFAULT SHARED CACHE *default-cache-name*

In a parallel sysplex environment, specifies the shared cache to be used for files associated with the segment. *Default-cache-name* must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system. Unless overridden at the file level, all files associated with the segment will use the named cache structure.

NULL

Removes any default shared cache from the segment.

DATA SHARING YES

At sites where data sharing is supported, specifies that areas within the segment are eligible to be concurrently updated by members of a data sharing group. This parameter may be overridden at the area level.

DATA SHARING NO

Specifies that areas within the segment are not eligible to be concurrently updated by members of a data sharing group. This parameter may be overridden at the area level.

file-override-specification

SHARED CACHE *cache-name*

In a parallel sysplex environment, specifies the shared cache to be used for the file. C*ache-name* must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.

NULL

Removes any shared cache association for the file.

DEFAULT

Specifies that the default cache specified for the segment will be used for the file.

area-override-specification

DATA SHARING NO

Specifies that the area is not eligible to be concurrently updated by members of a data sharing group.

DATA SHARING YES

At sites where data sharing is supported, specifies that the area is eligible to be concurrently updated by members of a data sharing group.

DATA SHARING DEFAULT

At sites where data sharing is supported, specifies that the data sharing attribute of the segment will apply to the area.

## Usage

***Specifying Data Sharing Attributes:*** Each data sharing group has an associated coupling facility lock structure. The first CA-IDMS system to become a member of the group, establishes the attributes of the lock structure. These attributes remain in effect until all members of the group have terminated normally. As long as any CA-IDMS system is either active or has failed and not yet been restarted, the existing lock structure attributes remain in effect.

Lock structure attributes include the number of lock entries and the maximum number of members. Both of these attributes affect the size requirements for the lock structure and should be chosen carefully.

***Specifying the number of lock table entries:*** The number of lock table entries determines the number of hash entries within the lock structure that will be used for managing locks. The higher the number of lock entries, the less chance that multiple resources will hash to the same lock table entry, a situation that results in increased overhead. However, the higher the number of lock table entries, the larger the size of the lock structure. See "Configuring the Coupling Facility" earlier in this chapter for more information on sizing the lock structure.

As a guideline, specify as the number of lock entries, the highest SYSLOCKS value of any CA-IDMS system that will be a member of the data sharing group. The number of lock entries will be rounded up to a power of 2.

***Specifying the maximum number of group members:*** The maximum number of group members determines the number of CA-IDMS systems that can be members of a data sharing group at any one time. If a CA-IDMS system terminates normally, it does not count as a group member for the purposes of this limit; however, if a CA-IDMS system terminates abnormally, it is still a member of the group until it is restarted and shut down normally. The higher the maximum member count, the more space is required in the lock structure. See "Configuring the Coupling Facility" earlier in this chapter for more information on sizing the lock structure.

The value specified may be overridden by CFRM policy. For more information, refer to the appropriate IBM manual. To determine the actual value in effect, use the DCMT DISPLAY DATA SHARING command.

***Specifying a shared cache:*** A shared cache is a structure defined within a coupling facility that allows data stored in the cache to be shared by multiple CA-IDMS systems. Assigning a file to a shared cache allows CA-IDMS systems to use the cache as a shared buffer.

Files may be assigned to a shared cache whether or not their associated areas are designated for data sharing. However, if an area is designated for data sharing, all of its associated files must be assigned to a shared cache.

***Specifying the connectivity loss option:*** The ON CONNECTIVITY LOSS parameter enables a site to specify what action a data sharing member should take in the event that connectivity to the coupling facility is lost or a failure in the coupling facility is detected. Specifying ABEND, directs the system to abend immediately; specifying NOABEND, directs the system to remain active as long as possible. By specifying NOABEND, it is possible for a member to remain active servicing requests for non-shared areas. It will not be possible to shut down the system normally.

## Using DCMT Commands

New DCMT commands provide the ability to:

- Change the default shared cache or the connectivity loss setting for an IDMS system. See DCMT VARY DATA SHARING in "New and Revised DCMT Commands."
- Enable or disable data sharing for an area or for all areas in a segment. See DCMT VARY AREA and DCMT VARY SEGMENT in "New and Revised DCMT Commands."

Existing DCMT commands can be used to dynamically specify the shared cache for a file, for all files associated with an area and for all files in a segment. See the DCMT VARY FILE, DCMT VARY AREA, and DCMT VARY SEGMENT in "New and Revised DCMT Commands."

Additional DCMT commands display information about data sharing. Refer to "Monitoring Data Sharing Groups" later in this chapter for information about these commands.

## Impact on the Runtime System

In order to provide the ability to share update access across CA-IDMS systems, several changes were made to the runtime system. This section describes differences in lock management and deadlock detection.

### Inter-CV-Interest in an Area

Inter-CV-interest applies to shared areas. It denotes a state in which the area is being accessed as shared by:

- At least one group member with a status of UPDATE, and
- More than one group member with a status of RETRIEVAL or UPDATE. Members accessing an area in TRANSIENT RETRIEVAL have no impact on inter-CV-interest.

Conflict for the area (and the records and pages in the area) can only occur if there is inter-CV-interest in the area. This is significant because if there is no inter-CV-interest in an area, the overhead associated with controlling access to it is reduced.

Whether or not there is inter-CV-interest in an area is indicated on the output from a DCMT DISPLAY AREA command. See "New and Revised DCMT Commands" for an example.

## Lock Management

Physical area locks control access to areas between CVs and between CVs and local mode applications. In Release 15.0, the management of physical area locks has been enhanced to allow concurrent update access among members of a data sharing group.

In prior releases, logical locks controlled access to individual records and areas by transactions executing within the same CV. In Release 15.0, this type of locking is referred to as *transaction locking*. Transaction locking controls access to records and areas by transactions executing within a single CV or within a single data sharing group.

A page lock is a new type of logical lock that is used within a data sharing group to protect the contents of a database page while it resides in a member's buffer pool.

For more information on CA-IDMS lock management in prior releases, refer to the chapter on Lock Management in the CA-IDMS Database Administration manual. The remainder of this section describes the differences introduced by Release 15.0.

## Physical Area Locking

For a non-shared area, a CV places a physical lock on the first page of an area when it opens the area for update. This prevents other CVs and applications executing in local mode from concurrently accessing the area in update mode.

For a shared area, the first member of a data sharing group to open the area for update places a physical lock on the area. Subsequent members opening the area for update are aware that they are not the first member to do so by examining information recorded in the coupling facility list structure associated with the data sharing group. Consequently, they expect that the physical area lock is set and don't consider its presence an error that otherwise would force the area offline.

Similarly the last member of a data sharing group to close a shared area accessed in update mode removes the physical area lock and updates the list structure accordingly.

## Transaction Locking

Transaction locks control access to individual records and areas. The basic locking scheme used in prior releases is still used in Release 15.0: the resource types are the same, the lock modes are the same and locks are acquired and released at the same times. However, if data sharing is in effect, there are two significant differences:

- Global locks are used to control inter-member access
- An additional level is introduced in the locking hierarchy

### Global Transaction Locks

Global transaction locks are used to control inter-member access to records and areas. Whenever a transaction places a lock on a shared area or on a record that resides in a shared area and there is inter-CV-interest in that area, global locks are used to ensure that no other transaction in the data sharing group is accessing the same resource in a conflicting mode.

Global locking relies on a coupling facility lock structure in order to record and manage global locks. Global locks are acquired by the CA-IDMS lock manager whenever a transaction places a lock on a resource and a sufficiently strong global lock is not already held by that CV. Global locks are retained until no transaction within a CV requires a lock of that strength, at which point the global lock may be released, downgraded or retained, depending on the resource type and whether or not there is contention for the resource between group members.

Global transaction locks are not acquired if there is no inter-CV-interest in an area. If inter-CV-interest begins because another member accesses the area in a potentially conflicting mode, global transaction locks will be acquired by every sharing member in which a transaction holds a local lock on the area or any of its records.

### New Locking Hierarchy

CA-IDMS has traditionally used a two-level locking hierarchy: area and record. Briefly summarized, this means that before placing a lock on a record, a transaction must place a lock on the area in which the record resides. Depending on the mode of the area lock, it may be possible to avoid placing locks on individual records within the area.

In release 15.0, if the area is designated for data sharing, the locking hierarchy expands to three levels: area, proxy, and record. Before a lock is placed on a record in a shared area, a transaction must hold a lock on a proxy that represents the record and before this can be done, it must hold a lock on the area in which the record resides.

A proxy represents a page of records. So before a lock can be placed on a record residing on a specific page in an area, a lock must first be placed on the proxy for the record's page.

A proxy can be locked in one of two modes: Share or Exclusive. A transaction must hold at least a share proxy lock before it can place a share or null (notify) lock on a record represented by the proxy. Similarly, a transaction must hold an exclusive proxy lock before it can place an exclusive lock on a record represented by the proxy.

An exclusive proxy lock held by one transaction does not prohibit access by another transaction. Instead the purpose of proxy locks is to detect inter-CV contention for resources and to eliminate the use of global record locks where possible. As long as all members holding a lock on a proxy hold it in share mode, there is no contention for resources on the page and no need to globally lock individual records on that page. However, if at least two members hold a lock on a proxy and at least one of those is an exclusive lock, then there is possible contention for individual records, necessitating the use of global record locks to control access.

The acquisition and management of proxy locks is done automatically by the CA-IDMS lock manager. Application programs do not need to be altered to exploit the new lock hierarchy. However, database administrators need to be aware of their existence and their impact on recovery and resource utilization.

## Page Locking

Within a data sharing group, page locks are used to protect database pages while they reside in a member's local buffer pool. Page locks are only placed on pages of areas that are designated for data sharing and only if there is inter-CV-interest in the area.

The coupling facility lock structure associated with the data sharing group is used to record and manage global page locks, just as is done for global transaction locks. And just as a proxy represents all of the records on the page, it also represents the page itself. Therefore, proxy locks reduce the need to acquire and release global page locks each time a page is moved into and out of the buffer pool.

Before a database page is read into the buffer pool, an exclusive or shared lock is placed on that page, depending on whether or not the active transaction intends to update the page. Once the lock is acquired, no other group member may place a conflicting lock on the page until the first member relinquishes its lock. This means that no other sharing member may update the page contents while another member has it locked and no other sharing member may read the page contents while another member has it locked exclusively. Page locks are held until another group member wants access to the page in a conflicting mode. Before an exclusive page lock can be released on an updated page, the page is written to the disk and to the shared cache.

## Deadlock Detection and Resolution

Deadlocks occur when there is unresolvable contention between multiple requestors for resources. Resources can be DC/UCF system resources (such as queues) or database resources (such as areas and records.)

In a single CV environment, deadlocks are detected by searching for stalled tasks at specified time intervals, identifying the resources on which they are waiting and the tasks holding those resources. This information is used to determine which tasks, if any, are deadlocked. Deadlocks are resolved by selecting tasks to be cancelled until the deadlock is eliminated. A user exit can be used to select a victim. If no user exit is provided, the task running for the shorter period of time or with the lesser priority is canceled.

Within a data sharing group, the tasks in contention for a resource may be executing on different group members. This introduces the potential for a global deadlock. In Release 15.0, a global deadlock detection mechanism has been added to resolve deadlocks occurring across group members.

A global deadlock is possible if at least one stalled task is waiting on a global resource. In a potential global deadlock situation, each member passes information to the one acting as the global deadlock manager. The global deadlock manager examines the information gathered from the other members and determines which tasks, if any, are deadlocked. User exits are invoked, to assist in selecting a victim task. If these exits are not provided, the task running for the shortest period of time or with the lowest priority is designated as the victim. Once the victim is determined, the member on which the victim is executing is directed to cancel the task.

### New User Exits

Two new user exits are used in selecting a victim in a global deadlock situation:

- **Exit #35** is invoked when a group member is collecting information about a stalled task in order to send it to the global deadlock manager. The exit provides the opportunity for a site to collect additional information that may be relevant to the victim selection process.

- **Exit #36** is invoked by the global deadlock manager when a victim is being selected. Its function is similar to the current exit #30, but it is passed different parameters. Instead of being passed the DCE addresses of two deadlocked tasks, it is passed a pair of parameters for each task, one of which is the information collected by exit #35. In this way, site-specific criteria can be used in selecting a victim even though the deadlocked tasks may be executing on a group member that is different than that of the global deadlock manager.

For details on coding these exits, refer to Appendix B, "New and Revised User Exits".

## Recovery Considerations

Sharing data in a multiple CV environment introduces the need for new recovery procedures to ensure the integrity of the database. The following sections address the types of failures that can occur, the impact they have in a shared environment and procedures to recover data.

## Member Failure

When a member of a data sharing group fails, recovery is typically effected by restarting the system and allowing warmstart to recover in the normal way. The primary data sharing consideration is that the system must be restarted using the same group and member names that were in effect at the time of failure.

Additionally, it is important to restart the system as soon as possible, since other members are prohibited from accessing resources that need to be recovered by the failing member. Refer to "Accessing Unrecovered Data" later in this chapter for more information on how the system deals with attempts to access unrecovered data.

If warmstart fails, manual recovery must be used to restore the database to a valid state. For more information about the impact of data sharing on manual recovery, refer to "Manual Recovery" later in this chapter.

## Coupling Facility Failures

The following types of failures relating to the coupling facility can occur:

- Loss of connectivity to a coupling facility structure
- Failure of a coupling facility structure
- An XES processing error

### Loss of Connectivity

Loss of connectivity to a coupling facility structure occurs when a connector (in this case a CA-IDMS system) can no longer communicate with the coupling facility in which the structure resides. Loss of connectivity may be the result of operator commands or hardware failures.

If connectivity is lost to either the list or lock structure associated with a data sharing group, the CA-IDMS system will either abnormally terminate or remain active so that requests that do not require access to the affected structure can be serviced. The action taken will depend on the connectivity loss option that is in effect for the system. If the system remains active, tasks that require access to the list or lock structure will be abnormally terminated.

### Structure Failure

Structure failure indicates that a structure residing in the coupling facility has been damaged. Damage may be due to hardware failures or XES processing errors.

CA-IDMS treats a failure of the lock or list structure associated with a data sharing group as a loss of connectivity to that structure. Depending on the connectivity loss setting in effect, the system will either abnormally terminate or remain active so that requests that do not require access to the affected structure can be serviced. If the system remains active, tasks that require access to the damaged structure will be abnormally terminated.

### XES Processing Error

An XES processing error will result in the abnormal termination of any CA-IDMS system that detects this condition. Contact IBM for the appropriate response. If there is any doubt as to the integrity of the lock and list structures associated with the data sharing group, follow the steps outlined below, deleting both the list and lock structures if necessary.

### Responding to Coupling Facility Failures

An error accessing the list or lock structure associated with a data sharing group, will result in the abnormal termination of every member that detects the problem. Termination may occur immediately, if the connectivity loss option indicates ABEND, or it may be deferred until an attempt is made to shut down the system. In either case, the system cannot be shutdown normally, because it cannot successfully disconnect from at least one of either the list or lock structures.

If the failure is due to a loss in connectivity, simply restart the failing CA-IDMS system(s) after taking appropriate action to restore connectivity. If the loss in connectivity is due to operator commands, issue the necessary commands to restore connectivity before restarting the system. If the problem is due to a hardware failure and another OS/390 image has connectivity to the coupling facility, the failed CA-IDMS systems can be restarted there. If no such OS/390 image exists, take one of the following actions:

- Correct the hardware problem and restart the systems.
- If an alternate coupling facility can be accessed, treat the loss in connectivity as a structure failure and respond as outlined below. Recreate the list and lock structures on the alternate coupling facility before restarting the systems.
- Undertake manual recovery to roll out only the transactions active at the time of failure. Initialize the journal files and restart one of the CA-IDMS systems as a stand-alone system (i.e., without data sharing enabled).

To recover from a failure of the list or lock structure, the following steps should be taken:

- Terminate all remaining systems in the data sharing group. If they won't shutdown, cancel them.
- Delete all connections to the failed structure and delete the failed structure using SETXCF FORCE commands.
- Restart all members that did not terminate normally. This will result in a group restart situation. For more information on restarting all members of a group, see Group Restart later in this chapter.

## Manual Recovery

Certain situations, such as an I/O error, necessitate the use of manual recovery to restore the database to a valid state. Manual recovery may also be used to restore the database to an earlier point in time, perhaps due to application errors.

There are two manual recovery scenarios. The first is used in place of warmstart to roll out uncommitted changes that were in effect at the time of the system failure. It is accomplished through the use of the ROLLBACK utility statement with the ACTIVE option and no STOP TIME. The second involves the use of the ROLLFORWARD or ROLLBACK utility statements with any option other than ACTIVE.

Data sharing impacts both of these recovery scenarios.

### Rollback Active

When using manual recovery in place of warmstart, the ROLLBACK utility statement is executed with the ACTIVE option and without a STOP TIME.

In a data sharing environment, it is possible to use this technique to recover a member in the event that warmstart fails. Before executing the rollback statement ensure that:

- All shared areas that were being updated by the failed member are varied offline or to transient retrieval in all active members of the data sharing group
- The journal files of all members to be recovered are offloaded with the ALL option
- If more than one member is to be manually recovered, the rollback operations are serialized.

Failure to take these steps may result in database corruption.

When all failed members have been recovered, their journal files should be initialized. The failed members should then be restarted in order to release the locks that were being held on the recovered resources.

Instead of running individual rollback operations, it is also possible to rollback active transactions using a merged journal file as input to the ROLLBACK utility. For more information on this technique, refer to "Merging Archive Files" later in this chapter.

### Other Recovery Scenarios

All other recovery scenarios that involve the use of ROLLFORWARD or ROLLBACK require that journal images for the files or areas being recovered be processed in sequence. In a data sharing environment, this becomes more difficult because the journal images for a shared area are contained on archive files associated with multiple IDMS systems. These images must be processed in chronological sequence in order for the recovery operation to be successful.

Release 15.0 provides a new MERGE ARCHIVE utility statement to create a file of journal images from multiple members in the sequence needed by the recovery utilities (ROLLFORWARD, ROLLBACK, and EXTRACT JOURNAL).

Depending on which options are to be used in the recovery operation, the journal images from multiple members may or may not require merging before executing a recovery utility. A merged file can always be used as input to a recovery utility, but under certain conditions, it is not required.

MERGE ARCHIVE must be used to merge the journal images of multiple data sharing group members before those images are processed by:

- ROLLFORWARD or ROLLBACK utility statements that specify the SEQUENTIAL option.
- ROLLFORWARD, ROLLBACK, or EXTRACT JOURNAL utility statements that specify both the ALL and STOP TIME options.

Provided that the above restrictions are met, the following are valid approaches to recovering a shared database.

- The ROLLFORWARD or ROLLBACK utility statement can be executed with the SORTED option using the archive files of all group members as input.
- The archive journal files of all group members can be processed by the EXTRACT JOURNAL utility statement and the output files concatenated as input to ROLLFORWARD
- The archive journal files from all group members that have updated the affected portion of the database can be merged prior to executing the ROLLFORWARD, ROLLBACK, or EXTRACT JOURNAL statement.

Each of these techniques is discussed below in more detail. No matter which technique is used, ensure the following:

- Before executing the recovery utility, quiesce the areas being recovered in all active group members, either by shutting down the systems or varying the areas offline or to transient retrieval
- Ensure that all necessary journal images are used for the recovery process

Sorted Rollforward/Rollback

To execute ROLLFORWARD or ROLLBACK without premerging the journal files, it must be executed using the SORTED option and without the combined ALL and STOP TIME options. The archive journal files of all group members that have updated the database since the last backup or since the last quiesce point must be concatenated as input. The archive files from each member must be concatenated in the sequence in which they were created. The order in which the files from different members are concatenated is not important.

Extract Journal

The EXTRACT JOURNAL utility statement is used to reduce the amount of time it takes to rollforward a restored portion of a database. In order to use this technique in a data sharing environment without premerging the journal files of different members, it must be executed without the combined ALL and STOP TIME options. The archive journal files from all group members that contain images for the portion of the database being recovered, must be processed by EXTRACT JOURNAL although not all need to be processed in a single execution.

If including archive files from multiple group members they can be concatenated in any order; however, archive files from a single member must be concatenated in the sequence in which they were created.

The output files produced by all executions of EXTRACT JOURNAL since the backup was taken must be concatenated as input to ROLLFORWARD. The extract files must be concatenated in the sequence in which the journal images for each member were created. All journal images for a given member must be in sequence, but the images across multiple members need not be.

Merging Archive Files

In a data sharing environment, if neither of the above techniques is used, the archive journal files of all group members must be merged together into a single file before executing the ROLLFORWARD, ROLLBACK, or EXTRACT JOURNAL utility statement. The new MERGE ARCHIVE utility statement is used to do the merge.

## Merge Archive Utility

This utility is used to merge the archived journal files of data sharing group members that are sharing update access to data. The output file created by the merge utility can be used as input to the ROLLFORWARD, ROLLBACK, EXTRACT JOURNAL, and MERGE ARCHIVE utility statements.

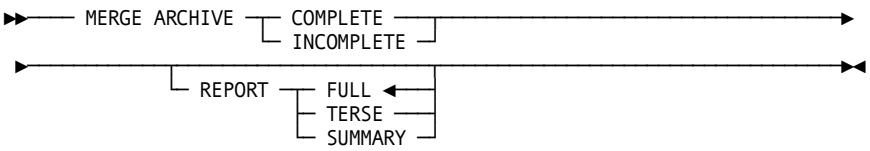
The merge utility also produces a report identifying global quiesce points within the set of merged journal images.

Under certain circumstances, this utility must be used if members of a data sharing group are sharing update access to data. It may be used in place of FIX ARCHIVE in a non-data sharing environment to merge several archive journal files from a single IDMS system. For more information on when the use of this utility is mandated, refer to "Manual Recovery" earlier in this chapter.

### Authorization

To merge archived journal files, you must hold the USE privilege on the DMCL.

### Syntax




### Parameters

COMPLETE

Indicates that all journal files that contain images needed for recovery have been included as input. ABRT checkpoint records will be added to the merged output file for all transactions still active at the end of the process. The output file may then be used as input to the ROLLBACK, ROLLFORWARD, or EXTRACT JOURNAL utility functions.

INCOMPLETE

Indicates that only a subset of the journal files have been included as input. The journal files are merged, but no ABRT checkpoints are generated. The output file may only be used as input to an EXTRACT JOURNAL or a subsequent MERGE ARCHIVE. You should not use the output file as input to ROLLBACK or ROLLFORWARD.

REPORT

Specifies the amount of detail that is to appear on the report produced by the merge utility.

FULL

Specifies that all details are to be reported. This includes for every transaction: checkpoints, database statistics and area usage. In addition, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

TERSE

Indicates that only transaction checkpoints and summary information is produced.

SUMMARY

Indicates that only final summary information is produced.

## Usage

*Processing flow*: MERGE ARCHIVE uses two input files: SYS001 and JRNM01 and one output file: SYS002. It proceeds by sorting the contents of SYS001 in chronological sequence and then merging the results of the sort with the contents of JRNM01. The resulting merged journal images are then processed and written to SYS002. Control records are also written to SYS002 indicating the range of images for each member that are included on the merged file and an indication of whether or not the output file was created with the COMPLETE option.

*Input files*: SYS001 is used to supply input that is not in chronological sequence. Typically, archive files from one or more data sharing members are concatenated together as input to SYS001, although it is also possible to include one or more merged files as input. The order of concatenation is not relevant.

JRNM01 is used to supply a single merged archive file. Multiple files cannot be concatenated as input to JRNM01. If no merged archive file exists, then JRNM01 should be specified as dummy.

*Incremental merging*: To minimize recovery time, journal files can be merged periodically and the output from each merge operation used as input to a subsequent MERGE ARCHIVE.

When merging journal files incrementally, specify the INCOMPLETE option on every MERGE ARCHIVE execution except the final one. The final merge operation before executing a ROLLFORWARD or ROLLBACK utility statement must specify the COMPLETE option.

*Using disk files*: It is possible to use disk files for merged journal files. This can be beneficial for incremental merging, since all intermediate merged files can reside on disk. Only the final merged file (the one created with the COMPLETE option) may need to be written to tape so that ROLLFORWARD or ROLLBACK can process it.

*Incomplete transactions:* If a transaction is encountered whose initial checkpoint record (BGIN) is not contained on the input file being processed, a warning message is written that will result in a return code of 4. This is not necessarily an error, since missing journal images can be merged at a later time; however, the missing journal records may need to be provided before the merged file can be used for recovery purposes. For more information, refer to the ROLLFORWARD or ROLLBACK utility commands.

## JCL Considerations

When submitting a MERGE JOURNAL statement through the batch command facility, in addition to the standard JCL required for the batch command facility, you must also include statements to define:

- SYS001 to point to the concatenated set of archived journal files and/or merged journal files.
- If the concatenated files have different block sizes, you must specify the following DCB parameter on the first file in the concatenation list:
- DCB=(RECFM=VB,BLKSIZE=nnnn)
- Where nnnn is greater than or equal to (4 + the largest block size of any file in the concatenation list).
- JRNM01 to point to a single merged journal file. If no such file exists, JRNM01 must be specified as DUMMY.
- SYS002 to point to the merged output file. The output file will have the block size that is specified for the archive journal file in the DMCL used for the merge.
- Any sort work files needed by your local sort

Refer to the appropriate chapter for your operating system in the *CA-IDMS Utilities manual* for generic JCL to execute the batch command facility.

## Example

The following statement directs the MERGE ARCHIVE utility to create a merged output file of all archived journal records and to write ABRT checkpoint records for any transactions still active when all input has been processed.

```
merge archive complete;
```

## Sample Output

The following is output generated after submitting a MERGE ARCHIVE statement to the batch command facility.

```
MERGE ARCHIVE   COMPLETE REPORT TERSE   ;
 NODE SYSTEM72 RU_ID         46 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  2000-03-02-04.34.55.920431
GLBQU
 NODE SYSTEM74 RU_ID         42 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  2000-03-02-04.34.55.930616
GLBQU
 NODE SYSTEM74 RU_ID         43 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  1   BGIN  2000-03-02-04.34.55.932905
 NODE SYSTEM74 RU_ID         45 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  2   BGIN  2000-03-02-04.34.55.998525
 NODE SYSTEM72 RU_ID         46 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  2000-03-02-04.34.56.785356
 NODE SYSTEM72 RU_ID         50 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  2000-03-02-04.34.57.062785
 NODE SYSTEM72 RU_ID         51 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  1   BGIN  2000-03-02-04.34.57.137212
 NODE SYSTEM72 RU_ID         52 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  2   BGIN  2000-03-02-04.34.57.148888
 NODE SYSTEM72 RU_ID         53 PGM_ID  DBCRUPD   QUIESCE LEVELS  4  UPD  3   BGIN  2000-03-02-04.34.57.152170
 NODE SYSTEM74 RU_ID         42 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  2   ENDJ  2000-03-02-04.34.57.206724
 NODE SYSTEM73 RU_ID         49 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  0   BGIN  2000-03-02-04.34.57.429637
...
 NODE SYSTEM73 RU_ID       1499 PGM_ID  DBCRUPD   QUIESCE LEVELS  6  UPD  6   ABRT  2000-03-02-04.41.20.714545
 NODE SYSTEM73 RU_ID       1458 PGM_ID  DBCRUPD   QUIESCE LEVELS  5  UPD  5   ENDJ  2000-03-02-04.41.20.729313
 NODE SYSTEM73 RU_ID       1359 PGM_ID  DBCRUPD   QUIESCE LEVELS  4  UPD  4   ABRT  2000-03-02-04.41.21.070481
 NODE SYSTEM74 RU_ID       1226 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  2000-03-02-04.41.21.096618
 NODE SYSTEM73 RU_ID       1472 PGM_ID  DBCRUPD   QUIESCE LEVELS  3  UPD  3   ENDJ  2000-03-02-04.41.21.293535
 NODE SYSTEM73 RU_ID       1498 PGM_ID  DBCRUPD   QUIESCE LEVELS  2  UPD  2   ABRT  2000-03-02-04.41.26.665860
 NODE SYSTEM73 RU_ID       1448 PGM_ID  DBCRUPD   QUIESCE LEVELS  1  UPD  1   ABRT  2000-03-02-04.41.26.807159
 NODE SYSTEM73 RU_ID       1392 PGM_ID  DBCRUPD   QUIESCE LEVELS  0  UPD  0   ENDJ  2000-03-02-04.41.26.820245  GLBQ


    ACTIVE PROGRAMS AT STOP TIME WERE:
   NONE


   DATABASE IN QUIESCE AT END OF FILE
   DATABASE IN UPDATE QUIESCE AT END OF FILE

   DATA BASE MAY NOT NEED TO BE RECOVERED

   SYS001 BLOCK COUNT       40  RECORD COUNT     6194
   JRNM01 BLOCK COUNT      342  RECORD COUNT    52235
   SYS002 BLOCK COUNT      382  RECORD COUNT    58429
```

## Group Restart

Group restart occurs when an inconsistency is detected during startup of a group member. CA-IDMS uses an XCF group and coupling facility lock and list structures in its data sharing support. Each of these maintains a status for each member of the group. Group restart is necessary if there are inconsistencies in these statuses or if either the lock or list structure doesn't exist when it should.

The purpose of group restart is to rebuild the lock and list structures from information in the journal files of previously failed members. The need for group restart is detected by a group member during startup. The first group member detecting the need for group restart becomes the restart coordinator. It is the coordinator's responsibility to monitor the progress of the restart process and direct the actions of other members.

In order to complete group restart, all failed members must be restarted. Members that had previously shut down normally may also be started during group restart, but this is not necessary. The restart coordinator will display messages on the JES log showing the progress of the restart and indicating which members still need to be restarted. Once all failed members known to the coordinator have been restarted, a message will be sent to the operator to confirm that all failed members have been restarted. It is the operator's responsibility to ensure that this is true before responding positively. Failure to include an abended member in the restart process can lead to corrupted data, since uncommitted updates by such a member have neither been rolled out nor the affected records locked to prevent access by other members.

During group restart, each previously failed member will update the list structure using information contained in its journal files. This information reflects the status of shared areas being processed by this member at the time of failure. The member then proceeds with its normal warmstart process to rollout incomplete transactions. Any transaction that cannot be rolled out will be restarted and appropriate locks acquired to protect the unrecovered data. Once this process is complete, startup is paused until all failed members have been restarted and reached this point in their processing. When this state is achieved, the restart coordinator, after confirmation from the operator, informs the other members that group restart is complete. Members then complete their startup process.

If group restart is interrupted because of a system failure, it can be restarted simply by restarting the failed systems. The procedures to follow after other types of failures will depend on the nature of the error.

## Accessing Unrecovered Data

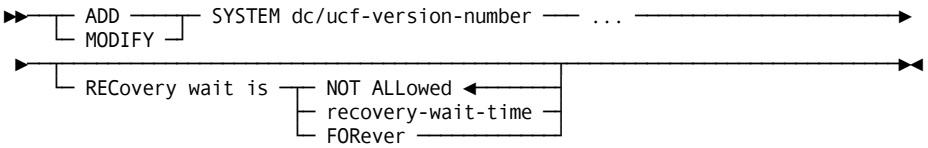
When a group member fails after it has made changes to a shared area and before those changes have been committed or rolled out, locks prevent access to the unrecovered data until the failing member is restarted. These locks might prevent access to an individual record, an entire page of records or an entire area, depending on the resource type and mode of the lock held by the failing member. It is therefore important to restart the failed member as soon as possible.

If a transaction on another system attempts to access unrecovered data, it can either wait for the failed member to recover or it can abort. The choice of actions is determined by the RECOVERY WAIT setting. This value is initially established in the system definition and can be dynamically changed using a DCMT command. A new user exit can override the recovery wait action for each task attempting to wait on an unrecovered resource.

## Recovery Wait Sysgen Parameter

The following parameter of the SYSTEM statement is new for Release 15.0. It determines what action should be taken if a task attempts to access a resource that needs to be recovered by a failed member of a data sharing group.

### Syntax




### Parameters

`RECovery wait is`

Specifies the amount of time the system is to permit a task to wait for a resource to be recovered by a failed data sharing group member before abnormally terminating the task.

*`recovery-wait-time`*

Specifies the recovery wait time in seconds. *Recovery-wait-time* must be an integer in the range 1 through 32,767. A value of zero is treated as if NOT ALLOWED was specified.

`NOT ALLowed`

Directs the system to immediately cancel the task. NOT ALLOWED is the default.

`FORever`

Directs the system to permit a task to wait indefinitely.

### Usage

*Changing the Recovery Wait Setting*: Use the DCMT DISPLAY TIME system task to display the current value for recovery wait. Use DCMT VARY TIME to dynamically change the value. Refer to "New and Revised DCMT Commands" for more information.

## Recovery Wait User Exit

Release 15.0 provides a recovery wait user exit to override the recovery wait setting for an executing task. User exit 37 is invoked whenever a task is about to wait on a resource that requires recovery by a failed data sharing member.

User exit 37 is passed a single parameter that contains the resource id that the task was attempting to lock and flags indicating its resource type. The exit indicates whether the task should wait or be cancelled and if it should wait, the length of time it should wait.

For details on coding this exit, refer to Appendix B, "New and Revised User Exits".

# Sharing Queues and Enqueued Resources

Data sharing groups provide the ability to share DC queues and enqueued resources between members of the group. This enables online tasks executing in different group members to communicate with one another just as if they were executing within the same CA-IDMS system. Each of these facilities is described below.

## Sharing Queues

In order to share queues between group members, the system queue area (DDLDCRUN) must be shared for update between members. Just as for any other area, the queue area is shared for update by designating it as such in the DMCL or through a DCMT command. A queue area may be shared by all or a subset of the group's members. There can be only one shared queue area per data sharing group. For more information on enabling data sharing for an area, refer to "Sharing Update Access to Data" earlier in this chapter.

If a CA-IDMS system is using a shared queue area, then all queues in that queue area are assumed to be shared except those specified as local in an exception table. A shared queue can be accessed by tasks executing on any group member sharing the queue area. There is only one shared queue with a given name in a queue area. Conversely, local queues are queues that can be accessed by only a single group member. There can be as many local queues with a given name as there are members sharing the queue area. Local queues are automatically qualified by their associated member name.

No changes are needed in application programs accessing queues in a shared queue area, regardless of whether they are local or global. The same queue commands (GET QUEUE, PUT QUEUE, etc.) are used.

Queue-initiated tasks are supported for shared queues, just as for non-shared queues. The member that causes the queue threshold to be exceeded is the one on which the queue-initiated task is executed.

### Designating Queues as Local

If a queue area is shared, then by default all queues in that queue area are shared. To specify a queue as local, add an entry to the LCLQUEUE module using the #LCLRES macro and reassemble LCLQUEUE. LCLQUEUE is distributed with one entry for the RHDCSETTIMETASKS queue. This entry should always be included in any reassembly of the module.

The queue name of the local queue is specified in the RESNAME parameter of the #LCLRES macro. Queues can be referenced generically by specifying an asterisk (*) as the last character of the RESNAME value. A queue name specified in this way indicates that all queues whose name matches that of the specified name (excluding the asterisk) are local.

The following example designates MYQUEUE as a local queue by adding an entry to the LCLQUEUE module:

```
#LCLRES TYPE=INITIAL,RESTYPE=QUEUE
#LCLRES RESNAME='RHDCSETIMETASKS'
#LCLRES RESNAME='MYQUEUE'
#LCLRES TYPE=FINAL
END
```

### Switching Queue Scope

If there is a need to switch a queue from being shared to local or vice versa, take the following steps:

- Process and delete all existing queue entries
- Use the QUED task to delete the queue
- Create a new LCLQUEUE module adding or removing entries as necessary
- Issue DCMT VARY NUCLEUS on every group member that shares the queue area to bring in a new copy of the LCLQUEUE module

### Impact of Shared Queues

Shared queues not only allow applications executing on different group members to communicate with one another, but they also impact the runtime system in the following ways:

- Report queues are shared globally by all members sharing the queue area. This means that reports can be printed on any group member that has a printer defined for the report's class.
- Checkouts of maps and dialogs are global across all members that share the queue area. This means that if a dialog is checked out on one member of the group, it is protected from check out on another member of the same group. The integrity of the checkout is guaranteed only if dictionaries are referred to by the same DBNAME in all sharing group members.
- Messages that are to be sent to users whenever they sign on (using the SEND…ALWAYS command) will be sent regardless of which member the user signs on to, provided those members are sharing the queue area with the member on which the DCUF command was issued.

## Sharing Enqueued Resources

Within a data sharing group, all enqueued resources are global by default. This means that if a task enqueues a resource exclusively on one member, that resource is unavailable on all other members of the group.

Resources can be designated as local. Enqueues on local resources impact only the system on which the enqueue is issued.

No changes in application programs are necessary in order to enqueue global or local resources.

## Designating Resources as Local

To designate a resource as local, add an entry to the LCLENQDQ module using the #LCLRES macro. LCLENQDQ is distributed with three entries needed for internal CA-IDMS processing. These entries should always be included in any reassembly of the module.

The resource id of the local resource is specified in the RESNAME parameter of the #LCLRES macro. Generic resource names can be specified by coding an asterisk (*) as the last character of the RESNAME value. A generic resource name indicates that all resources whose id matches that of the specified name (excluding the asterisk) are local.

The following example designates any resource beginning with "MYRES" as a local resource by adding an entry to the LCLENQDQ module:

```
#LCLRES TYPE=INITIAL,RESTYPE=ENQDEQ
*
* The following resource-ids are used internally by the
* system, and need a local scope in a data sharing * group environment.
*
 #LCLRES RESNAME='RHDCD09I'
 #LCLRES RESNAME='RHDCTIMP'
 #LCLRES RESNAME='USERJRNL'
*
* Define other local resources here
*
 #LCLRES RESNAME='MYRES*'
*
        #LCLRES TYPE=FINAL
        END
```

## Switching Resource Scope

If there is a need to switch an enqueued resource from being global to local or vice versa, take the following steps:

- Create a new LCLENQDQ module adding or removing entries as necessary
- Issue DCMT VARY NUCLEUS on every group member to bring in a new copy of the LCLENQDQ module

## Defining Local Print Tasks

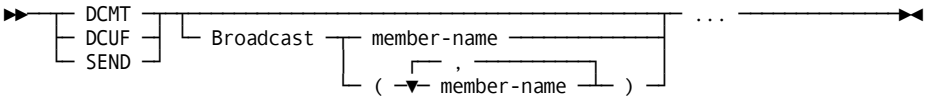
A local resource should be defined for every print task defined using the #UCFUFT macro. To do this, add an entry to the LCLENQDQ module specifying the PTID value in the #UCFUFT macro as the RESNAME value in the #LCLRES macro.

# Broadcasting Commands

Data sharing groups provide the ability to execute the same system task on all or a subset of group members by issuing a single request. This is supported for the following commands:

- All DCMT commands except those that require a user interaction. The commands for which it is not supported are DCMT SHUTDOWN, DCMT ABORT, DCMT VARY/DISPLAY NUCLEUS, and DCMT VARY DMCL.
- DCUF SHOW USER
- SEND

## Syntax




## Parameters

`Broadcast`

Directs CA-IDMS to execute the command on all or the specified set of members of the data sharing group.

*`member-name`*

Specifies a member on which CA-IDMS is to execute the command. If no member-name is specified the command is executed on all members of the data sharing group.

## Usage

*Authorization*: The issuing user must have the authority to execute the command on all members of the group to which it is directed. If the needed authority is not held on a member, the command will not execute on that member, but may on other members.

*Output*: The output from a broadcasted command is segmented by member. All output from one member will be displayed before that of another member. When broadcasting to all members, the output for the member on which the command is issued will be displayed first. Other member's output is identified by a header indicating the name of the member.

## Examples

The following are examples of broadcasted commands:

```
dcmt broadcast vary segment cust offline

dcuf b show users all

dcmt b (cust01, cust02) display dbname
```

The following is an example of the output resulting from a broadcasted command:

```
DCMT B V SEGMENT EMPDEMO OFFLINE
---------- Area -----------   Lock    Lo-Page    Hi-Page #Ret  #Upd #Tret #Ntfy
EMPDEMO.EMP-DEMO-REGION        Ofl      75001      75100    0     0     0     0
 Stamp: 1998-11-17-09.55.31.875826  Pg grp: 0      NoShare  NoICVI  NoPerm

EMPDEMO.INS-DEMO-REGION        Ofl      75101      75150    0     0     0     0
 Stamp: 1998-11-17-09.55.31.956231  Pg grp: 0      NoShare  NoICVI  NoPerm

EMPDEMO.ORG-DEMO-REGION        Ofl      75151      75200    0     0     0     0
 Stamp: 1998-11-17-09.55.31.887739  Pg grp: 0      NoShare  NoICVI  NoPerm

===> Output from group member SYSTEM73

---------- Area -----------   Lock    Lo-Page    Hi-Page #Ret  #Upd #Tret #Ntfy
EMPDEMO.EMP-DEMO-REGION        Ofl      75001      75100    0     0     0     0
 Stamp: 1997-08-07-14.58.14.855461  Pg grp: 0      NoShare  NoICVI  NoPerm

EMPDEMO.INS-DEMO-REGION        Ofl      75101      75150    0     0     0     0
 Stamp: 1997-08-07-14.58.14.896650  Pg grp: 0      NoShare  NoICVI  NoPerm

EMPDEMO.ORG-DEMO-REGION        Ofl      75151      75200    0     0     0     0
 Stamp: 1997-08-07-14.58.14.874287  Pg grp: 0      NoShare  NoICVI  NoPerm
```

# Monitoring Data Sharing Groups

The following facilities can be used to monitor a data sharing group:

- DCMT commands
- Perfmon
- Journal reports

## Monitoring Through DCMT Commands

The following commands enable monitoring of various aspects of a data sharing group:

- DCMT DISPLAY AREA
- DCMT DISPLAY DATA SHARING
- DCMT DISPLAY LOCK STATISTICS

## DCMT DISPLAY AREA

This command has been enhanced to show the sharability state of an area and whether or not there is inter-CV-interest in the area.

Accessing a shared area for which there is inter-CV-interest will result in higher overhead because global locking must be used to control access to the area by members of the data sharing group.

Accessing a shared area for which there is no inter-CV-interest will incur only slightly more overhead than would be needed if the area were not shared.

## DCMT DISPLAY DATA SHARING

This new command displays the following types of information:

- A list of group members and their status
- Statistics associated with accessing the data sharing group's list and lock structures
- Information regarding the available space in the lock structure
- Statistics on XCF messages used for inter-member communication

Detailed descriptions of the output of this command are provided in "New and Revised DCMT Commands."

### Member Status

Each member of a data sharing group has a member state that is assigned by XCF and a user state that is assigned by CA-IDMS.

The XCF member states that may be associated with CA-IDMS systems are:

- Active – indicating that the member is currently executing
- Failed – indicating that the member has terminated abnormally

The user states that may be associated with CA-IDMS systems are:

- Initial – indicating that startup is in progress for the member
- Recovering – indicating that the member is in the process of recovering from a prior abnormal termination
- Ready – indicating that the member has completed recovery and is ready to open the database system
- Active – indicating that startup is complete
- Quiescing – indicating that the system is in the process of closing the database system
- Quiesced – indicating that the database system has been closed

### Monitoring Available Lock Structure Space

Record data entries are stored in the coupling facility lock structure to record information about exclusive global transaction locks. The more concurrently held exclusive locks, the more space is needed in the lock structure.

The output of the DCMT DISPLAY DATA SHARING command allows you to monitor the available space in the lock structure by displaying the maximum number of record data entries that can be stored in the lock structure, the current number that are in use, the highest number that were ever used and the number of times the lock structure encountered a short-on-storage condition.

As the lock structure becomes full, CA-IDMS will release exclusive locks on proxies, if possible, to relieve the short-on-storage condition; however this is undesirable because it means it increases the overhead associated with global locking. Furthermore, even releasing all unused proxy locks may not free up enough space in the lock structure. If there is not enough space to store a record data entry when one is required, the task requesting the lock will fail.

The amount of space in the lock structure can be increased while group members remain active by using the SETXCF START ALTER command, provided the coupling facility in which the lock structure resides contains sufficient free space. If it doesn't, the lock structure must be reallocated. To do this all members in the data sharing group must be shutdown and the CFRM policy changed.

## DCMT DISPLAY LOCK STATISTICS

The output from this command displays information about local and global locks acquired to control access to transaction resources.

Some points to note:

- The ratio of global resource lock requests to local lock requests is a measure of contention for resources between members. If there is no contention, then this ratio will be small since the only global resource locks acquired will be for areas and in an active system in which areas are always readied in a shared mode, global area locks will generally be retained once they are acquired.
- The ratio of the number of waits to the number of global requests is also a measure of contention. This contention may be due to resource conflicts or other factors, such as channel contention, or false contention caused by synonyms when hashing to the lock table. If this ratio is high, use operating system tools to determine the nature of the contention. False contention can be reduced by increasing the number of entries in the lock table. See "Enabling Data Sharing" earlier in this chapter for information on specifying the number of lock table entries.

## Monitoring Through Performance Monitor

Performance Monitor has been altered to collect and report on information associated with a data sharing group. This new capability is available through the Interval Monitor. There are new online display screens and new Performance Monitor reports available.

To view the new online display screens select Sysplex Menu from the Interval Monitor Menu screen.

### Sysplex Menu (PF23)

```
PM-R15.0 SYSTEM71            Computer Associates Intl. V71            00.010 12:17:51.04
CMD-->                                                                      Window : 02
    Dtl   Hist  Description            Dtl   Hist  Description
    _     _     DBGroup                _     _     Shared Cache
    _     _     Data Sharing Lock      _     _     Data Sharing List
    _     _     Data Sharing Member
```

#### Menu description

The Sysplex Menu is a sub-menu of the Interval Monitor. It incorporates two items that were previously on the main menu and makes available three new items associated with data sharing.

The Sysplex Menu allows selection of the following displays:

| Screen Name | Display |
|---|---|
| DBGroup Detail | Information for the current interval, showing statistics related to each DBGroup that can process dynamically routed database sessions. Additionally, each DBGroup can be selected to show the distribution of the DBGroup requests processed by the different server nodes (DBGroup's Node screen). |
| DBGroup History | One line per interval for the DBGroup wait category |
| Data Sharing Lock Detail | Information for the current interval, showing statistics related to each type of global lock acquired in a data sharing environment. |

| Screen Name | Display |
|---|---|
| Data Sharing Lock History | One line per interval for the Data Sharing Lock wait category |
| Data Sharing Member Detail | Information for the current interval, showing statistics related to each member of this system's data sharing group. |
| Data Sharing Member History | One line per interval for the Data Sharing Member wait category |
| Shared Cache Detail | Information for the current interval, showing statistics for each shared cache active in the interval. Additionally, each Shared Cache can be selected to show the same information by files (Shared Cache Files Detail Screen). |
| Shared Cache History | One line per interval for the Shared Cache wait category |
| Data Sharing List Detail | Information for the current interval, showing statistics related to list in the list structure associated with this system's data sharing group. |
| Data Sharing List History | One line per interval for the Data Sharing List wait category |

## Data Sharing Lock Detail

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                        Window : 03
ResType          Obtains      Alters    Releases       Waits    WaitTime     AvgWait
LmgrResource           1           0           1           0     .0000S      .0000S
Phys.Page            134           0         134           0     .0000S      .0000S
GlobalDeadLk           0           0           0           0     .0000S      .0000S
LmgrProxy              1           0           1           0     .0000S      .0000S
EnqDeq                 0           0           0           0     .0000S      .0000S
AreaList               1           0           1           0     .0000S      .0000S
FileList               3           0           3           0     .0000S      .0000S
GlobalQueue            1           0           1           0     .0000S      .0000S
- - - -
HighWait      ContEx     NotifEx
 .0000S           0           0
 .0000S           0           0
 .0000S           0           0
 .0000S           0           0
```

```
 .0000S          0          0
 .0000S          0          0
 .0000S          0          0
 .0000S          0          0
```

### Screen description

The Data Sharing Lock Detail screen displays statistics related to acquiring global locks. The screen includes one line for each type of global resource for which locks can be acquired.

### What to look for

Look for excessive average wait time.

## Data Sharing Lock History

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                        Window : 03
  Start    Waits     Wait      Avg           .2        .4        .6        .8         1
  Time               Time      Wait  ---------|---------|---------|---------|---------0
_ 13:43       0     .0000S    .0000S
_ 13:45       1     .0015S    .0015S
_ 13:50       4     39.72S    9.93S  ------------------------------------------------
_ 13:55       0     .0000S    .0000S
_ 14:00       0     .0000S    .0000S
```

### Screen description

The Data Sharing Lock History screen displays each interval being tracked. For Each interval the screen shows a total count and time for global lock waits. This screen also shows the average wait time for the interval. The average wait time is displayed numerically and in graph form.

### Using this screen

To request the Detail screen for an interval, type any other nonblock blank character to the left of the interval for which the detail is required and press [Enter] or move the cursor to an interval line and press [PF9].

### What to look for

Use the graphic display to determine intervals with higher than average waits.

## Data Sharing Member Detail

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                         Window : 03

Member   Member   Current  Prior    ReplyMsg ReplyMsg  TestMsg  TestMsg SyncStamp SyncStamp
name     state    CVstate  CVstate      Sent Received     Sent Received      Sent  Received
SYSTEM71 Active   Active   Ready           0        0        0        0         0         0
SYSTEM72 Active   Active   Ready           0        0        0        0         0         0
- - - -
SyncStamp GlblDeadLk DCMTUFSEND DCMTUFSEND
 Received       Sent   Received       Sent
        0          0          0          0
        0          0          0          0

PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                         Window : 03

Member   AreaFileVa AreaFileVa QueueMsg QueueMsg ProgramMsg ProgramMsg
name           Sent   Received     Sent Received       Sent   Received
SYSTEM71          0          0        0        0          0          0
SYSTEM72          0          0        0        0          0          0
```

### Screen description

The Data Sharing Member Detail screen displays each data sharing member that was a member of this system's data sharing group during the interval. The screen includes one line for each member showing its member state, its current and prior CV states and the number of messages sent from this system to the given member and from the member to this system.

### What to look for

Look for excessively high numbers of messages.

## Data Sharing Member History

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                         Window : 03
  Start    Waits     Wait      Avg           .2        .4        .6        .8         1
  Time               Time      Wait  ---------|---------|---------|---------|---------|
_ 12:25        0    .0000S    .0000S
_ 12:30        0    .0000S    .0000S
_ 12:35        0    .0000S    .0000S
_ 12:40        0    .0000S    .0000S
_ 12:45        0    .0000S    .0000S
_ 12:50        0    .0000S    .0000S
_ 12:55        0    .0000S    .0000S
_ 13:00        0    .0000S    .0000S
_ 13:05        0    .0000S    .0000S
_ 13:10        0    .0000S    .0000S
_ 13:15        0    .0000S    .0000S
_ 13:20        0    .0000S    .0000S
```

### Screen description

The Data Sharing Member History screen displays each interval being tracked. For each interval the screen shows a total count and time for waits on messages. This screen also shows the average wait time for the interval. The average wait time is displayed numerically and in graph form.

### Using this screen

To request the Detail screen for an interval, type any other nonblock blank character to the left of the interval for which the detail is required and press [Enter] or move the cursor to an interval line and press [PF9].

### What to look for

Use the graphic display to determine intervals with higher than average waits.

## Data Sharing List Detail

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                         Window : 03

ListName              Reads        Writes       Deletes         Waits       WaitTime
AreaList                 14             9             0             7         .0028S
FileList                 15            21             0             0         .0000S
QueueList                 1             0             0             1         .0058S
- - - -

AvgWait       HighWait
 .0004S         .0007S
 .0000S         .0000S
 .0058S         .0058S
```

### Screen description

The Data Sharing List Detail screen displays each list in the list structure associated with this system's data sharing group that had activity during the interval. The screen includes one line for each list showing its name and statistics for the various types of accesses to the list.

### What to look for

Look for excessively high average wait times.

## Data Sharing List History

```
PM-R15.0 SYSTEM71             Computer Associates Intl. V71             00.010 12:17:51.04
CMD-->                                                                         Window : 03
  Start    Waits     Wait      Avg           .2        .4        .6        .8         1
  Time               Time      Wait  ---------|---------|---------|---------|---------|
_ 07:51        0    .0000S    .0000S
_ 07:55        0    .0000S    .0000S
```

```
_ 08:00       0    .0000S    .0000S
_ 08:05       3    2.07S     .6910S ---------------------------------
_ 08:10       0    .0000S    .0000S
```

### Screen description

The Data Sharing List History screen displays each interval being tracked. For each interval the screen shows a total count and time for waits on requests to access lists in the list structure. This screen also shows the average wait time for the interval. The average wait time is displayed numerically and in graph form.

### Using this screen

To request the Detail screen for an interval, type any other nonblock blank character to the left of the interval for which the detail is required and press [Enter] or move the cursor to an interval line and press [PF9].

### What to look for

Use the graphic display to determine intervals with higher than average waits.

## Monitoring Through Journal Reports

The following journal reports have been modified to show the nodename of the system that created the journal image. In a data sharing environment, the nodename is the same as the system's membername.

- Journal report 1 – Transaction Summary
- Journal report 2 – Program Termination Statistics
- Journal report 3 – Program I/O Statistics
- Journal report 5 – Detail Area/Transaction
- Journal report 6 – Detail Program/Area
- Journal report 8 – Formatted Record Dump

In addition, journal report 8 has been modified to show the time when the journal image was created. This time is based on GMT (Greenwich Mean Time). This time, in conjunction with a journal sequence number, is used to sort and merge journal images, both for images created by a single system and across members in a data sharing environment.

# Dynamic Routing Enhancements

Release 14.0 introduced the ability to dynamically route database sessions between CA-IDMS systems through the use of DBGroups. This feature has been enhanced in Release 15.0 in the following ways:

- Fewer structures are needed in the coupling facility to support dynamic routing
- The backend nodes to which requests can be routed need not be defined in a frontend system's resource table
- If data sharing is used to share update access to data, update transactions may now be dynamically routed

## Fewer Coupling Facility Structures

Prior to Release 15.0, the implementation of a DBGroup for the purpose of dynamically routing database requests required the definition of two list structures in the coupling facility.

In Release 15.0, each DBGroup requires only a single coupling facility list structure. The name of this list structure is:

CAIDMS*DBGroupname*

where *DBGroupname* is the name of the DBGroup.

Use the following formula for estimating the size of the list structure:

Size = (*HSM* * *FE* * 256) + 256K

where

- *HSM* is the highest value of MAXIMUM TASKS in the SYSTEM statement for each frontend system.
- *FE* is the number of frontend systems.

Because of the change in list structures, it is not possible to dynamically route requests from a pre-Release 15.0 system to a Release 15.0 system or vice versa.

## Dynamic Backend Determination

In Release 15.0, it is no longer necessary to define the nodes that can service a dynamically routed database request in the frontend system's node table. Not only does this reduce the administrative overhead associated with defining DBGroups, but it allows new back-ends to be defined, or existing backends to be moved within the sysplex without requiring changes to the system definition of each frontend system.

If a node definition for the target backend system exists within the frontend system's node table, the access method specified in the table will be used for communication between the two systems. If, however, no matching node entry exists, the access method will be chosen as follows:

- If the two systems are executing on the same OS/390 image and their system definitions specify the same SVC number, then the SVC access method will be used.
- If the backend system has a VTAM line driver, then the VTAM access method will be used.
- If the backend system has a CCI line driver, then the CCI access method will be used.

### Dynamically Routing Update Transactions

Because data sharing permits more than one CA-IDMS system to update a database concurrently, update transactions can now be dynamically routed to any system that has update access to the target data. This also means that there is no need to distinguish between retrieval and update transactions for the purpose of dynamic routing.

## Shared Cache Enhancements

Release 14.0 introduced shared cache to enable database buffers to be shared between CA-IDMS systems. This feature continues to exist in Release 15.0 and can be used either in conjunction with, or independently of, data sharing.

To accommodate data sharing, the implementation of shared cache was changed in the following ways:

- The coupling facility list structure associated with shared cache is no longer required
- The AVAILABLE option is no longer supported
- The minimum cache item size has been reduced from 2K to 256 bytes

### Fewer Coupling Facility Structures

Prior to Release 15.0, the use of shared cache required that a list structure named ZIDMSCACHE0001 be defined in the coupling facility. This list structure is no longer used. The only coupling facility structures that are needed for shared cache are the cache structures that contain database pages.

### AVAILABLE Option Removal

AVAILABLE is no longer supported as a shared cache option either in the DMCL file override or in DCMT commands. When dynamically assigning a shared cache to a file, the assignment must be made on each system for which shared cache is to be used. Within a data sharing group, this can be done using the new broadcast capability for DCMT commands. See "Broadcasting Commands" earlier in this chapter for more information on this feature.

### Item Size Reduction

The minimum size of a cache item has been reduced in 15.0. While this does not require redefinition of a cache structure, it does mean that a shared cache cannot be simultaneously in use by both pre- and post-release 15.0 systems.

## Cloned System Enhancements

Release 14.0 introduced the ability to start multiple CA-IDMS systems using a single system definition. This feature was referred to as cloned systems.

In Release 15.0, cloned system support has been enhanced in the following ways:

- If a clone is a member of a data sharing group, it can update any area that is designated as shared.
- The nodename is the membername for any clone that is also a member of a data sharing group.

### Updating through a Clone

Prior to Release 15.0, all database areas accessed by a clone were forced to a status of RETRIEVAL, with the exception of the system log and queue areas. In Release 15.0,this restriction has been lifted for areas that are designated as shared, if the clone is a member of a data sharing group.

This allows the convenience of sharing a single system definition to be extended to updating systems.

### Pre-determined Nodename

In order to accommodate data sharing, the nodename for a system that is both a data sharing group member and a clone is set to be the system's membername.

The SVC number and VTAM APPLID are still established dynamically for cloned systems.

Chapter

4

# Other Release 15.0 Enhancements

## Overview

Release 15.0 provides a number of features other than those related to sysplex exploitation. This chapter describes the features that provide non-stop processing, usability or performance improvements.

## Non-Stop Processing Enhancements

### Dynamic Storage Pool

Release 15.0 provides the ability to dynamically add or increase the size of an XA storage pool. This enables the system to be configured to handle increasing workloads or a changing workload that requires additional storage without recycling the system.

To use this feature, take the following steps:

- Add a new XA storage pool or increase the size of an existing XA storage pool using the system generation compiler
- Generate the system definition
- Use the DCMT DISPLAY SYSGEN REFRESH command with the STORAGE POOL option to display the pending storage pool changes
- Use the DCMT VARY SYSGEN REFRESH command with the STORAGE POOL option to make the change effective

The ability to add a new storage pool or increase the size of an existing storage pool may be limited by the amount of storage available to the CA-IDMS system. Other types of changes to XA storage pools in the system definition, such as deleting a storage pool, or reducing its size will be ignored by the DCMT VARY SYSGEN REFRESH commands.

For more information on the above DCMT commands refer to Appendix A, "New and Revised DCMT Commands."

## Dynamic Program Pool

Release 15.0 provides the ability to dynamically add or increase the size of either the XA Program Pool or the XA Reentrant Pool. This enables the system to be configured to handle increasing workloads or a changing workload that requires additional XA program pool space without recycling the system.

To use this feature, take the following steps:

- Add or increase the size of either the XA Program Pool or the XA Reentrant Pool using the system generation compiler
- Generate the system definition
- Use the DCMT DISPLAY SYSGEN REFRESH command with the PROGRAM POOL option to display the pending program pool changes
- Use the DCMT VARY SYSGEN REFRESH command with the PROGRAM POOL option to make a change effective

The ability to add a new program pool or increase the size of an existing program pool may be limited by the amount of storage available to the CA-IDMS system. Other types of changes to program pools in the system definition, such as deleting a program pool, or reducing its size will be ignored by the DCMT VARY SYSGEN REFRESH commands.

For more information on the above DCMT commands refer to Appendix A, "New and Revised DCMT Commands."

## Dynamic External Wait Time for a Task

Release 15.0 allows the external wait time for a task to be changed dynamically. This feature permits a new external wait time to be established for an existing task definition or the external wait time to be specified for a dynamically defined task without recycling the CA-IDMS system.

To use this feature:

- To change the external wait time for an existing task, specify the new EXTERNAL WAIT parameter on the DCMT VARY TASK command
- To specify an external wait time for a new task, use the EXTERNAL WAIT parameter on the DCMT VARY DYNAMIC TASK command

## Dynamic Change in Multitasking Queue Depth

Release 15.0 provides the ability to dynamically change the multitasking queue depth. This enables the system to be tuned without recycling the CA-IDMS system.

To use this feature:

- To display the current multitasking queue depth, use the DCMT DISPLAY MT command.
- To change the multitasking queue depth, use the DCMT VARY MT command.

For more information on the above DCMT commands refer to Appendix A, "New and Revised DCMT Commands."

## Tolerance of Database I/O Errors

In prior releases, an I/O error on a database file encountered during warmstart would cause warmstart to fail, necessitating the use of manual recovery procedures before the system could be restarted.

In Release 15.0, warmstart is now tolerant of database I/O errors and other file-related errors that prevent recovery, such as an inconsistent data set name in a data sharing environment. If such an error occurs, warmstart will proceed to roll out as many uncommitted changes as it can. Transactions that are unable to be completely rolled out will be restarted and suspended. This allows the system to continue to execute while corrective action, such as restoring a damaged file, is taken. Once the problem has been corrected, a DCMT VARY FILE ACTIVE command must be issued for the file that encountered the error. This will cause suspended transactions to complete rollback.

A transaction that cannot be completely rolled out during warmstart, is referred to as an unrecovered transaction. No ABRT checkpoint is written for an unrecovered transaction during warmstart, because the transaction will be restarted during system startup. Restarting a transaction involves rebuilding the transaction control blocks and acquiring exclusive locks on all records updated by the transaction subsequent to its last commit. A restarted transaction will retain its original transaction identifier and can be displayed using any of the normal transaction display facilities such as DCMT commands or Performance Monitor screens.

All unrecovered transactions will be restarted under a single task.

When startup completes, a ROLLBACK will be issued for each unrecovered transaction. If the rollback is successful, the transaction terminates and rollback will be attempted for the next unrecovered transaction. If the rollback fails (likely due to the same condition that was encountered during warmstart), the transaction will be suspended until a DCMT VARY FILE ACTIVE command is issued. When the error has been corrected and the file varied active, the unrecovered transaction will complete rollback. This process will continue until all unrecovered transactions have been rolled out.

It is important to take corrective action as soon as possible if unrecovered transactions exist, since portions of the database are locked from access by other transactions until recovery is complete.

### Expanded System Counters

To support non-stop processing, the following counters have been increased from 2 to 4 bytes:

- The number of system tasks (STRTKSYS)
- The number of times the maximum task limit was reached (STRTKMAX)
- The number of times the system encountered a short-on-storage condition (STRTKSOS)

These counters are all located in the #STRDS dsect. Appropriate changes have been made to display the increased field sizes in DCMT commands and Performance Monitor.

### Restartable Deadlock Manager

Release 15.0 will now automatically restart the deadlock manager task in the event that it should abend. No action is necessary for this to happen; however if the deadlock manager fails consecutively more than 5 times, the system will abnormally terminate.

## Ease-of-Use Enhancements

### Installation Control of Storage Protection

Release 15.0 installation provides the ability for a site to select whether or not CA-IDMS supplied programs will be defined with storage protection enabled. This capability allows users to more easily tailor their system definitions to meet their requirements.

In prior releases, CA-IDMS supplied programs were initially defined with the PROTECT option, indicating that storage protection was enabled for the program. While this may be appropriate in some DC/UCF systems, such as test systems, it is generally not recommended for production systems due to the overhead associated with storage protection.

The Release 15.0 installation process allows the specification of which option (PROTECT or NOPROTECT) is to be used when defining CA-IDMS supplied programs. Furthermore, there are now two sets of source members containing task and program definitions for CA-supplied components: one set defines programs with the PROTECT option and the other with the NOPROTECT option. The appropriate set can be selected when creating new system dictionaries.

For details on the new installation parameter and the new source members, refer to the appropriate CA-IDMS Installation Guide for your operating system.

## Streamlined CICS Interface Installation

This feature simplifies the SMP/E installation of CA-IDMS by modifying the manner in which the IDMS interface and the various Transparencies for CICS are provided and installed, while continuing to allow the interface to be tailored to a particular environment.

The new interface is tailored using parameters specified during the installation process that are stored in an options module. The generated installation job stream will link this options module with the appropriate distributed object decks under SMP/E control, removing the compilation of the interface as part of the installation process. Subsequent maintenance of the CICS interface and transparencies will be accomplished by applying standard APARs through SMP/E.

To use the new interface, a site must be running CICS 4.1.0 or higher. Sites running earlier versions of CICS must continue to use the existing CICS interface which is delivered under the name INTCR141. Refer to the Release 14 documentation for instructions on how to generate a CICS interface using this macro.

The new CICS interface is tailored through the CICSOPT macro which the installation job stream assembles using parameters supplied during the installation process. The parameters for this macro are identical to those for the old IDMSINTC macro. For details, refer to Appendix C, “CICSOPT Macro.”

Users that have extensively altered the existing CICS interface through source changes will continue to have this ability since the new interface is delivered in source as well as object form. However, such users will be unable to use SMP/E for subsequent maintenance. If it is necessary to modify the new interface to meet the requirements of your site, the source is contained in two macros: IDMSINTC and IDMSESC.

## CICS Interface Enhancements

Release 15.0 provides the following functional enhancements to the CICS interface:

- TPNAME default
- Detection of CWADISP conflict
- Identification of the originating interface in messages
- Limits on the number of CICS users accessing IDMS through a particular interface
- Dynamic routing assistance
- OPTI exit support for SQL

### TPNAME Default

The TPNAME parameter identifies the CICS system in which an IDMS request originates. In Release 15.0, if this parameter is omitted during the assembly of the CICSOPT macro, it will default to the local CICS system id in which the interface is executing. This enables the same IDMS interface module to be used in multiple CICS systems.

### CWADISP Conflict

The CWADISP parameter identifies a fullword in the CICS CWA that is to be used by the IDMS interface. In Release 15.0, when the interface first starts, it will check to ensure that the value in this field is zero. A value other than zero indicates that the specified location is being used for some other purpose and will result in an interface abend with a code of K005.

### Interface Identification

All messages generated by the CICS interface now identify both the name of the interface module and its release number. This enables monitoring of the status of a particular interface and provides more information in the event of an error.

CICS interface messages now have the following format:

```
CA-IDMS <release-number> : <interface-module-name> : <message text>
```

## Limiting CICS Users

The number of users that can access IDMS through a given CICS interface can now be limited. The USERCNT parameter of the CICSOPT macro determines the maximum number of CICS users that can be accessing IDMS at any one time. This includes all active sessions, all suspended sessions, and all users that have not yet timed out. The default value for this parameter is 100. Refer to Appendix C, for a full description of the CICSOPT macro.

## Dynamic Routing Support

The CICS interface now supports OPTIXIT extensions and multiple SYSCTL files enabling users to implement dynamic routing for CICS applications accessing IDMS. This allows a site to route IDMS sessions to a set of backend CVs using the OPTIXIT to implement a routing strategy appropriate for their environment.

The new MAXCVNO parameter of the CICSOPT macro specifies the number of additional SYSCTL files that a particular interface will support. The value specified must be an integer between 0 and 9. A value of 0 indicates that only a single SYSCTL file is used. Values between 1 and 9 indicate that the specified number of additional files are processed. The DDNAMEs for the additional files are generated from the value of the SYSCTL parameter by replacing the first blank (or the last character if the SYSCTL value is 8 characters) with a value from 1 to the MAXCVNO value. For example, if the SYSCTL value is "PROD" and the MAXCVNO value is 2, the following SYSCTL DDNAMEs will be processed by the interface:

```
PROD
PROD1
PROD2
```

In addition to supporting multiple SYSCTL files, a new argument is passed to the OPTIXIT. Among other things, it contains the addresses of the OPTI control blocks that are generated from the additional SYSCTL files. With this information, the OPTIXIT may then select the backend CV to which a request is routed either based on the specifics of the request or on work load balancing algorithm implemented within the exit. For more information on the OPTIXIT, refer to Appendix B, "New and Revised User Exits". For more information on the CICSOPT macro, refer to Appendix C, "CICSOPT Macro".

### OPTI Exit Support for SQL

The Release 15.0 CICS interface provides a new user exit for selecting the backend CV to which an SQL session will be routed. The entry point of this new exit is OPTIQXIT. Many of the arguments passed to this exit are the same as those for the OPTIXIT, thus enabling it to take full advantage of the new dynamic routing support. The remaining parameters are specific to SQL and provide information about the application program making the request and in certain circumstances, the name of the dictionary to which the session is being connected.

For more information on this new exit, refer to OPTIQXIT in Appendix B, "New and Revised User Exits."

## Increased Number of Files

For OS/390 users the maximum number of files that can be accessed by a DC/UCF system has been increased. This allows access to an increased amount of data from a single CV without requiring changes to application programs.

Normally an OS/390 job step can access up to 3,273 files. CA-IDMS has extended this limit for a CV, to allow up to 10,000 files to be accessed using dynamic allocation and 3,273 files to be accessed using DD statements.

To exploit this feature, the DMCL associated with the system must be altered to increase the number of files that are included. If more that 3,273 files are to be accessed, the excess files must be defined for dynamic allocation.

**Note:** Since the maximum number of DD statements that can be associated with a job step is 3273, if the number of database files in a DMCL is close to or exceeds this limit, dynamic allocation should be used for all database files so that the limit will not prevent the use of DD statements to override dynamically allocated files when necessary.

In addition to changing the DMCL, it may also be necessary to reduce the CA-IDMS region size, both by adjusting the system definition and lowering the REGION parameter specified on the EXEC card in the startup JCL.

Increasing the number of files beyond the 3273 limit, has implications for manual recovery, since the increased limit is supported only for CVs and not local mode batch jobs such as utility executions. In order to perform manual recovery, it may be necessary to execute the ROLLBACK or ROLLFORWARD utility statement multiple times, recovering a subset of the areas, or segments in each execution.

## Dynamic Allocation Control

Release 15.0 provides the ability to control whether or not files will be dynamically allocated in a local mode job. By default, if dynamic allocation information is included in the DMCL used by a local mode job, then CA-IDMS will dynamically allocate a file unless it is overridden through the execution JCL. Unfortunately, this has occasionally resulted in access to unintended files when the JCL override was mistyped.

Release 15.0 allows a site to control whether dynamic allocation is used or not. This can be done in two complimentary ways: by establishing a site default and overriding the default using a SYSIDMS parameter.

Establishing a site default is done by creating a SYSIDMS options module and specifying a value for the new LOCAL_DYNAMIC_ALLOCATION parameter. By default, the value of this parameter is ON, meaning that dynamic allocation is used in local mode. Specifying a value of OFF directs CA-IDMS to not use dynamic allocation in local mode.

The site default can be overridden for an individual job step by specifying the LOCAL_DYNAMIC_ALLOCATION parameter in the SYSIDMS file associated with the job step.

For information in creating a SYSIDMS options module, refer to *CA-IDMS Database Administration – Volume 2*.

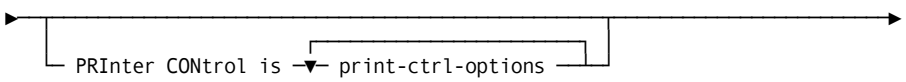
## Printer Formfeed Options

Release 15.0 allows the specification of printer formfeed options. Default options may be specified for the system and overridden for individual printers as necessary.

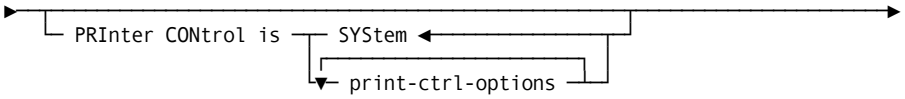
To use this feature, specify the PRINTER FORMFEED option on the SYSTEM statement of the system definition to establish default options for the system. Specify the PRINTER FORMFEED option on the LTERM statement to specify options for an individual printer.

### Syntax

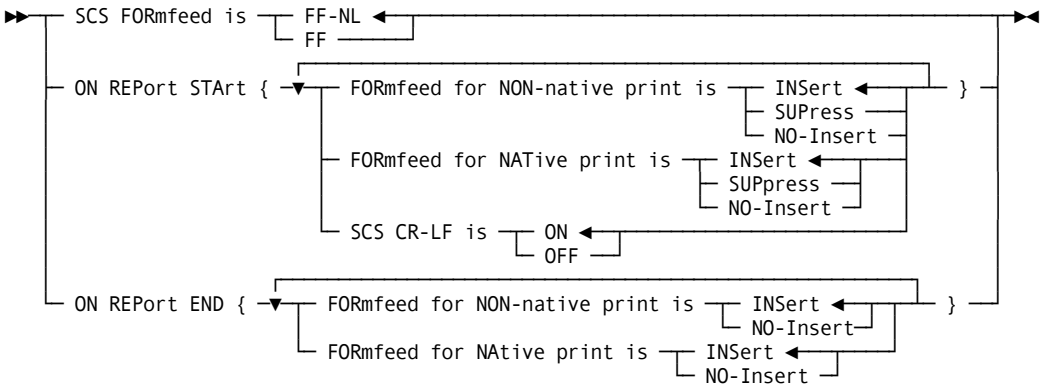
**SYSTEM statement:**



**LTERM statement:**




**Expansion of** print-ctrl-options**:**




## Parameters

If specified on the SYSTEM statement, the options apply to all printers defined for the system, unless overridden for an individual printer.

`SYStem`

Specifies that the default printer control options established for the system are to be used for this printer. SYStem is the default for the PRInter CONtrol parameter specified on the LTERM statement.

`SCS FORmfeed is`

Specifies the form feed sequence that will be sent to SCS devices. The options that may be specified are:

- FF-NL—Specifies that the form feed sequence is "FF" (form feed) followed by "NL" (new line). FF-NL is the default.
- FF—Specifies that the form feed sequence contains "FF" only.

`ON REPort STArt`

Specifies what happens when a report starts printing. The options that may be specified are:

- FORmfeed for NATive print is—Specifies the form feed processing for native mode print reports.

- FORmfeed for NON-Native print is—Specifies the form feed processing for non-native mode print reports.
- The parameters that may be specified for the above two FORmfeed options are:
  - INSert: a form feed will be inserted if not yet there.
  - SUPpress: the report will never start with a form feed, i.e., if the report starts with a form feed, it will be removed.
  - NO-Insert: no change to the report contents will be made.
- SCS CR-LF is—Suppresses the Carriage Return/Line Feed sequence at the beginning of a report transmitted to an SCS printer.

```
ON REPort END
```

Specifies what happens when a report finishes printing. The options that may be specified are:

- FORmfeed for NATive print is—Specifies the form feed processing for native mode print reports.
- FORmfeed for NON-Native print is—Specifies the form feed processing for non-native mode print reports.
- The parameters that may be specified for the above two FORmfeed options are:
  - INSert: a form feed will be inserted.
  - NO-Insert: no form feed will be inserted.

## Controlling SVCs from High-level Languages

In OS/390 environments, Release 15.0 enables control over which SVCs can be issued from COBOL and PL/I programs executing within a DC/UCF system. This provides an increased level of security and administrative control within CA-IDMS.

SVC screening is used to trap operating system SVC requests issued by high-level language runtime support modules. SVC screening enables programs written in COBOL and PL/I to execute under the control of a DC/UCF system. In order to allow more control over which SVCs are executed from high-level language programs, SVC screening now traps all SVCs issued by assembler routines linked with or called directly from COBOL or PL/I programs. By default, if the SVC is not supported by the CA-IDMS runtime system, the issuing task will abend with an abend code of T*xxx* where *xxx* is the number of the SVC that was issued.

A site may choose to allow additional SVCs to be issued from high-level languages by applying an optional APAR. Alternatively, the high-level language program can be changed to invoke the assembler routine through a TRANSFER CONTROL DML command. This latter approach, while requiring that the assembler program be defined to the DC/UCF system, allows any SVC to be issued. With either approach, the DBA's involvement increases the level of control over SVC invocations.

## DBKEY Stall Information

In Release 15.0, if a task is cancelled due to exceeding its stall time while waiting for a lock on a transaction resource (a dbkey or an area), then additional information will be displayed showing both what resource the task was waiting on and what tasks were holding the lock. This additional information may be used to identify application conflicts that reduce system throughput.

The following is an example of the messages that are generated for a stall on a dbkey:

```
DC001007 V74 T64 TASK:LOCK2 PROG:LOCKTEST WAITING FOR LTXNLOCK 00000008 0124FF01
DC001008 V74 T64 TXNID:44757 PROG:LOCKTEST SUBSCHEMA:EMPSS01 LINE#:30 MODE:S
DC001009 V74 T64 TSKID:63 TASK:LOCK PROG:LOCKTEST HOLDS LTXNLOCK 00000008 0124FF01
DC001010 V74 T64 TXNID:44756 PROG:LOCKTEST SUBSCHEMA:EMPSS01 MODE:X
```

## Mixed DBKEY Radixes within Page Group

A single page group may now encompass areas with different DBKEY radixes without running the risk of false contention.

In prior releases it was recommended that all areas within a page group have the same DBKEY radix. Failure to adhere to this guideline could have resulted in false contention for records when in fact no such contention existed. In Release 15.0, this is no longer an issue.

In order to eliminate false contention for records, the dbkey radix has been added to the lock resource id. The format of a lock resource id is now:

```
ppppffrr xxxxxxxx
```

Where:

```
    pppp: is the page group
      Ff: is '00' for a DBKEY
             '80' for an area
      rr: is the DBKEY radix
xxxxxxxx: is the DBKEY or page number
```

It is still the case that sets cannot be defined in which the owner and member are in areas with different DBKEY radixes. Furthermore, transactions attempting to access data with different radixes (even in the same page group), may require modifications similar to those for accessing data in mixed page groups.

## Permanent Area Status

The status of an area within a CA-IDMS system can now be set so that it is retained across system shutdowns. Prior releases provided the ability to retain an area status across abnormal system terminations by specifying ON WARMSTART MAINTAIN CURRENT STATUS for the area in the DMCL. Release 15.0 allows the status of an area to be retained across normal terminations as well.

To use this feature, specify the PERMANENT option on the DCMT VARY AREA/SEGMENT commands.

The status of an area that has been established as permanent will remain in effect until it is changed with another DCMT command or until the journal files are initialized.

To determine if an area's status has been established as permanent, issue the DCMT DISPLAY AREA/SEGMENT command. For more information on these commands, refer to "New and Revised DCMT Commands."

## VARY AREA Enhancements

Release 15.0 provides the following functional enhancements associated with varying the status of an area:

- An area can be varied to RETRIEVAL or TRANSIENT RETRIEVAL regardless of the existence of notify locks
- A new option directs CA-IDMS to force a vary operation by canceling tasks and user session that conflict with the vary
- Outstanding vary area commands can be cancelled

### Notify Locks and Varying Areas

Changes have been made in the way CA-IDMS deals with a vary request for an area that has outstanding notify locks.

- If the area status is changing to RETRIEVAL or TRANSIENT RETRIEVAL, the vary will occur regardless of whether or not notify locks exist for dbkeys in the area. In prior releases, the vary operation would wait until all notify locks had been released before completing the status change.
- If the area status is changing to UPDATE, the vary will occur immediately as in prior releases. However, since it is possible that the area was updated externally while it was in retrieval mode to this CV, in Release 15.0:
  - Subsequent tests of a notify lock that existed at the time of the vary will indicate that both prefix and data have been modified

  - The restoration of currencies associated with the area that were saved prior to the vary will result in a task abend if the area is readied in an update mode

- If the area status is changing to OFFLINE, the vary will wait until all notify locks have been released as in prior releases.

### Vary Area IMMEDIATE

When changing an area's status to RETREIVAL, TRANSIENT RETRIEVAL, or OFFLINE, a new IMMEDIATE option can be specified which directs CA-IDMS to cancel active tasks, terminate user sessions, and vary predefined system run units offline if they prevent the change in status. This forces the change to take place, regardless of what may be executing and can be used when operational circumstances necessitate it.

For more details, refer to Appendix A, "New and Revised DCMT Commands".

### Canceling Vary Area Operations

Vary area operations that are delayed due to conflicts with executing tasks or user sessions can be displayed using the new DCMT DISPLAY ID command and cancelled using the new DCMT VARY ID command.

To enable this functionality, each vary area operation that is changing the area status to RETRIEVAL, TRANSIENT RETRIEVAL, or OFFLINE is assigned a unique identifier which is either specified on the DCMT VARY AREA/SEGMENT command or is generated internally. The DCMT DISPLAY ID command lists all outstanding vary operations together with their identifiers. The DCMT VARY ID command can be used to cancel a specific vary operation or all operations whose identifier matches a wildcarded value.

For more details, refer to Appendix A, "New and Revised DCMT Commands".

## Area Quiesce

Release 15.0 provides a new DCMT command to quiesce database areas. Quiescing all or a portion of a database may be done as part of normal operations or in response to an emergency situation. The new command establishes a quiesce point and once established can initiate further processing through a user exit. A quiesce operation can automatically terminate itself, or can be cancelled explicitly.

## The Quiesce Operation

A quiesce operation is initiated by a DCMT QUIESCE command. A single quiesce operation can apply to:

- An individual area
- All areas matching a wildcarded name
- All areas in a specified segment
- All areas in segments included in a specified DBNAME.

Once a quiesce operation has been initiated, it will continue until a quiesce point has been established. A quiesce point is a point in time at which no transactions are accessing the target areas in update mode. In order to achieve a quiesce point, tasks attempting to access a target area in update mode for the first time, wait until the quiesce operation has completed. Tasks that are already updating a target area are either allowed to continue or are aborted depending on user-specified options. Similarly, transactions that are accessing a target area in update mode but are in a pseudo-conversational state are either allowed to continue or are forced to terminate (by having their resources deleted). Additionally, predefined system run units that conflict with a quiesce operation may be varied offline in order to reach a quiesce point.

Once a quiesce point has been established, a message identifying the current time is written to the log, a journal swap may be initiated and user exit 38 is invoked. The user exit can initiate further processing, such as a backup by submitting a job through the internal reader. The quiesce operation can then terminate automatically or remain active until explicitly terminated.

For a complete description of the DCMT QUIESCE command, refer to Appendix A, "New and Revised DCMT Commands."

## Monitoring a Quiesce Operation

When a quiesce operation is initiated, it must be assigned a unique identifier that distinguishes it from other quiesce operations that may be active. While the quiesce operation is in progress, its status can be monitored by issuing the new DCMT DISPLAY ID command. This command will either display the status of an individual quiesce operation or all quiesce operations in progress.

A quiesce operation can be terminated, either before or after the quiesce point has been established by issuing a DCMT VARY ID command.

For a complete description of these new commands, refer to Appendix A, "New and Revised DCMT Commands."

## Quiescing Areas in a Data Sharing Environment

If one or more areas to be quiesced are shared, the quiesce operation is distributed to all members of the data sharing group automatically. The member on which the quiesce operation originated becomes the coordinator of the quiesce operation. The coordinator is responsible for initiating the subordinate quiesce operations on the other members of the group, monitoring their progress and terminating the quiesce operation. The quiesce operation can only be terminated by the coordinator or through a DCMT VARY ID command issued on the coordinator. If the coordinator abends, the quiesce operation is automatically terminated.

In a data sharing environment, failed members may prevent the establishment of a quiesce point. If a failed member was updating a target area at the time of failure, the quiesce operation cannot complete until the failed member is restarted. In this situation, the coordinator will display operator messages every ten seconds indicating which failed members must be restarted in order to complete the quiesce operation.

If a new member is started while a quiesce operation is in progress, it is informed of the outstanding quiesce and will prevent tasks from updating the quiescing areas until the quiesce operation terminates.

## Quiesce User Exit

A new user exit (exit 38), is invoked when a quiesce point is reached. The exit is passed the quiesce identifier, an indication of what is being quiesced and a list of files that are impacted by the quiesce and their dataset names.

In a data sharing environment, the user exit is invoked only on the coordinator and not on the other members of the data sharing group.

The purpose of this exit is to allow further actions to be taken in response to the quiesce. For example, the exit could submit a batch job through the internal reader to backup the quiesced areas. Following are two possible approaches for exploiting this functionality.

**Approach 1**

User Exit

- Submits a job through the internal reader
- Returns control to IDMS indicating that the quiesce operation should continue

Batch Job

- Backs up the database (possibly using an "instantaneous" copy facility available with certain disk devices)

- Terminates the quiesce operation by issuing a DCMT VARY ID command through batch UCF

**Approach 2**

User Exit

- Submits a job through the internal reader
- Returns control to IDMS indicating that the quiesce operation should be terminated

Batch Job

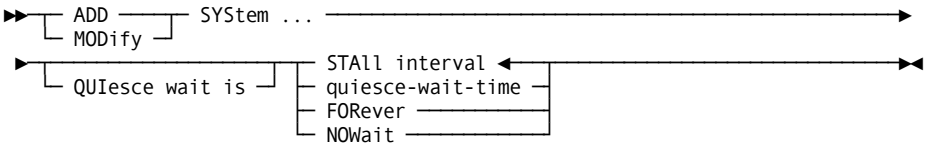
- Backs up the database (creating a hot-backup since the database is being updated while the backup is in progress)
- Initiates a second quiesce operation in order to limit the volume of journal images that must be processed by a sequential ROLLFORWARD should a recovery become necessary

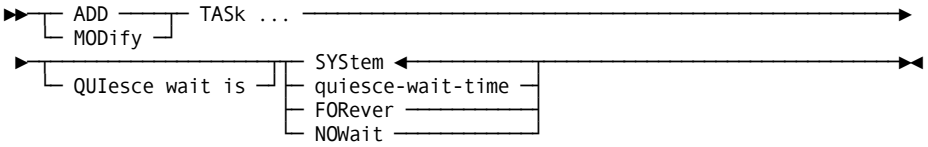
For a complete description of exit 38, refer to Appendix B, "New and Revised User Exits."

## Quiesce Wait Time

If a task must wait to gain update access to an area because the area is being quiesced, whether or not it waits and the amount of time it waits is determined by its quiesce wait time. The quiesce wait time for a task is initially established through the system definition and can be overridden at run time through DCMT VARY TIME and DCMT VARY TASK commands. The new sysgen parameters are described below. The new options on the DCMT commands are described in Appendix A, "New and Revised DCMT Commands."

## Syntax



### System Level Parameters

```
QUIesce wait is
```

Specifies whether or not the system permits a task to wait for a quiesce operation to terminate and if waiting is permitted, the amount of time the task will wait before being abnormally terminated.

- STAll interval—Specifies that the quiesce wait time for a task is the same as its stall interval. This is the default if no quiesce wait parameter is specified.
- *Quiesce-wait-time*—Specifies the quiesce wait interval in wall-clock seconds. Quiesce-wait-time must be an integer in the range 1 through 32,767.
- FORever—Directs the system not to terminate tasks based on quiesce wait time.
- NOWait—Specifies that tasks will not wait for quiesce operations to terminate and instead will receive an error indicating that an area is unavailable. For navigational DML requests, this will result in an error-status value of 'xx66'.

### Task Level Parameters

```
QUIesce wait is
```

Specifies whether or not the system permits the task to wait for a quiesce operation to terminate and if waiting is permitted, the amount of time the task will wait before being abnormally terminated.

- SYStem—Specifies that the quiesce wait time for the task is determined by the quiesce wait setting for the system. This is the default if no quiesce wait parameter is specified.
- *Quiesce-wait-time*—Specifies the quiesce wait interval in wall-clock seconds. Quiesce-wait-time must be an integer in the range 1 through 32,767. The specified value overrides the QUIESCE WAIT parameter of the current SYSTEM statement.
- FORever—Directs the system not to terminate the task based on a quiesce wait time. FORever overrides the QUIESCE WAIT parameter of the current SYSTEM statement.
- NOWait—Specifies the task will not wait for quiesce operations to terminate and instead will receive an error indicating that an area is unavailable. For navigational DML requests, this will result in an error-status value of 'xx66'. NOWait overrides the QUIESCE WAIT parameter of the current SYSTEM statement.

## Recovery Utility Enhancements

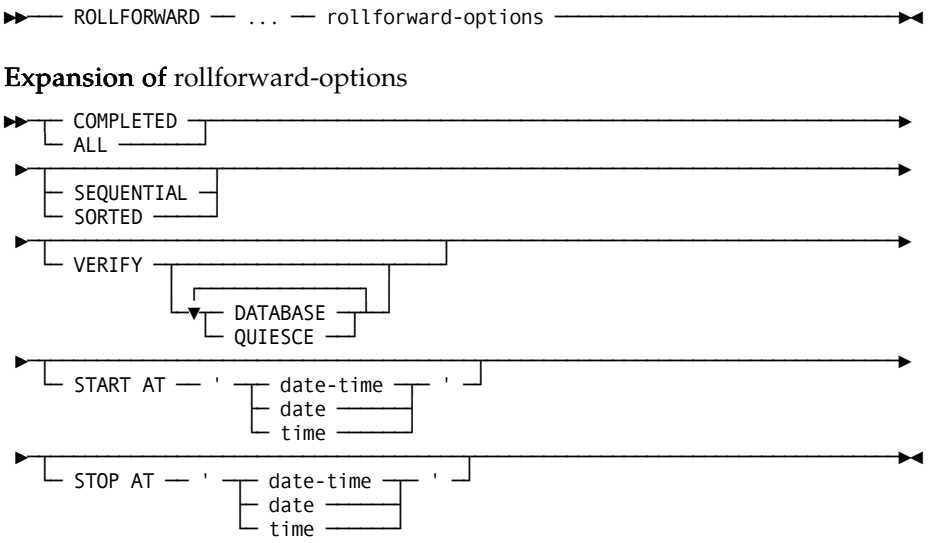
The 15.0 recovery utilities have been enhanced in the following ways:

- A start time can now be specified on the ROLLFORWARD and EXTRACT JOURNAL utility statements, enabling recovery to begin at a specific point in time rather than at the beginning of the archive file. This may save recovery time and also circumvent issues with aborted transactions active at the beginning of the archive file.
- A new verify option can be specified on the ROLLFORWARD and EXTRACT JOURNAL utility statements to ensure that recovery begins at a quiesce point for the target areas.
- A new parameter can be specified on the ROLLBACK utility statement indicating that the database was restored from a hot backup file. This enables the ROLLBACK utility to process the archive journal files created while a hot backup is in progress.

The changes to each of these utilities are described below.

### Rollforward Utility Statement

### Syntax



### Parameters

VERIFY

Indicates that the validity of the specified conditions should be checked. If the VERIFY option is not specified, no checking is performed. If VERIFY is specified without DATABASE or QUIESCE, DATABASE is the default.

- DATABASE—Specifies that before images are to be verified in the journal file before the after images are applied. If a before image in the journal file does not match the contents of the database, the roll forward operation will terminate with an error.

  By default, if database validity checking is not in effect, after images are copied back without first verifying the before images.
- QUIESCE—Specifies that the beginning of the recovery operation must coincide with a quiesce point for the portion of the database being recovered. If any recovery unit both updated the target areas and spans the start of the journal file, the operation will terminate with an error.

START AT

Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time will be applied. If any transaction that updated the portion of the database being recovered spans the start time, the operation will terminate with an error.

By default, processing starts at the beginning of the journal file.

- *date*—Specifies the date, in one of the following formats:
  - *yyyy-mm-dd*
  - *mm/dd/yyyy*
  - *dd.mm.yyyy*

In these formats, the following rules apply:

- Yyyy specifies the year. Yyyy must be an integer in the range 0001 through 9999. Leading zeros are optional.
- Mm specifies the month. Mm must be an integer in the range 01 through 12. Leading zeros are optional.
- Dd specifies the day within the month. Dd must be an integer in the range 01 through 31. Leading zeros are optional. The combined values of *yyyy*, *mm* and *dd* must represent a valid date. For example, 1988-02-29 is a valid date. 1989-02-29 is not.

- *date-time*—Specifies the date and time, in the following format:
  - yyyy-mm-dd-hh.mm.ss.ffffff

The rules for specifying the date component of *date-time* are the same as for *date* described above. The rules for specifying the time component of *date-time* are:

- *Hh* specifies the hour on a 24-hour clock. *Hh* must be an integer in the range 00 through 23. Leading zeros are optional.
- *Mm* specifies the number of minutes past the hour. *Mm* must be an integer in the range 00 through 59. Leading zeros are optional.
- *Ss* specifies the number of seconds past the minute. Ss must be an integer in the range 00 through 59. Leading zeros are optional.
- *Ffffff* specifies the number of millionths of a second past the specified second. *Ffffff* is optional; if you include it, it must be an integer in the range 0000000 through 999999. Trailing zeros are optional.

- time—Specifies the time, in one of the following formats:
  - hh.mm.ss
  - hh:mm:ss

The rules for specifying *time* are the same as those listed for the time component of *date-time* above. When *time* is specified, the date defaults to the current date.

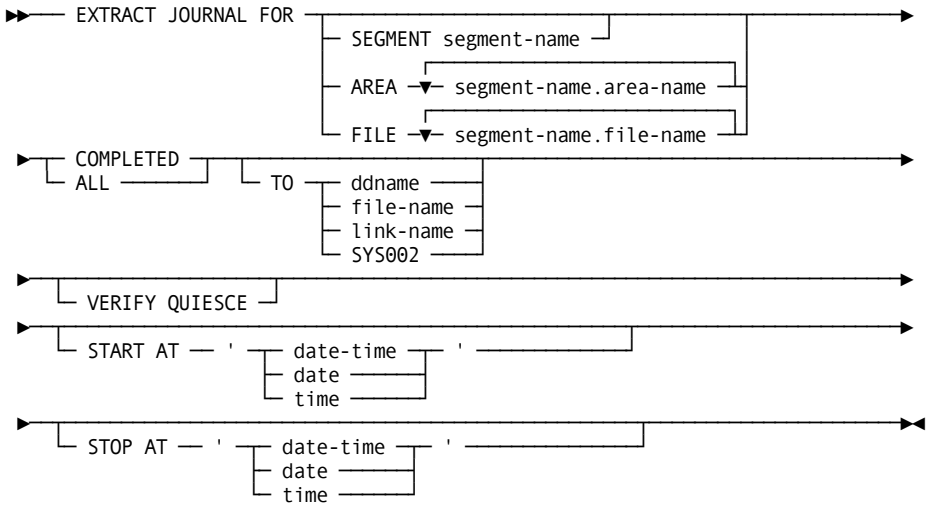
## Usage

*Verifying quiesce points:* A quiesce point is defined to be a point in time for which no recovery unit that is active has a before or after image on the journal file for a portion of the database being recovered. A recovery unit is represented by the journal images starting at BGIN or COMT checkpoint and ending with a COMT, ABRT, or ENDJ checkpoint.

If a start time is specified, it must coincide with a quiesce point, otherwise the roll forward operation will terminate with an error. If no start time is specified, you can indicate whether or not the start of the journal file should be a quiesce point through the use of the VERIFY QUIESCE option. If specified, the roll forward operation will terminate with an error unless the start of the journal file coincides with a quiesce point.

## Extract Journal Utility Statement

### Syntax




### Parameters

VERIFY QUIESCE

Specifies that the beginning of the extract process must coincide with a quiesce point for the specified portion of the database. If any recovery unit both updated the specified portion of the database and spans the start of the journal file, the operation will terminate with an error.

START AT

Specifies that the operation should start only after reaching a specified date and time in the journal file. Only images for transactions that begin after the specified start time will be written to the extract file. If any transaction that updated the specified portion of the database spans the start time, the operation will terminate with an error. By default, processing starts at the beginning of the journal file.

Refer to the START AT option of the Rollforward Utility Statement above for a description of the *date*, *date-time* and *time* parameter values.

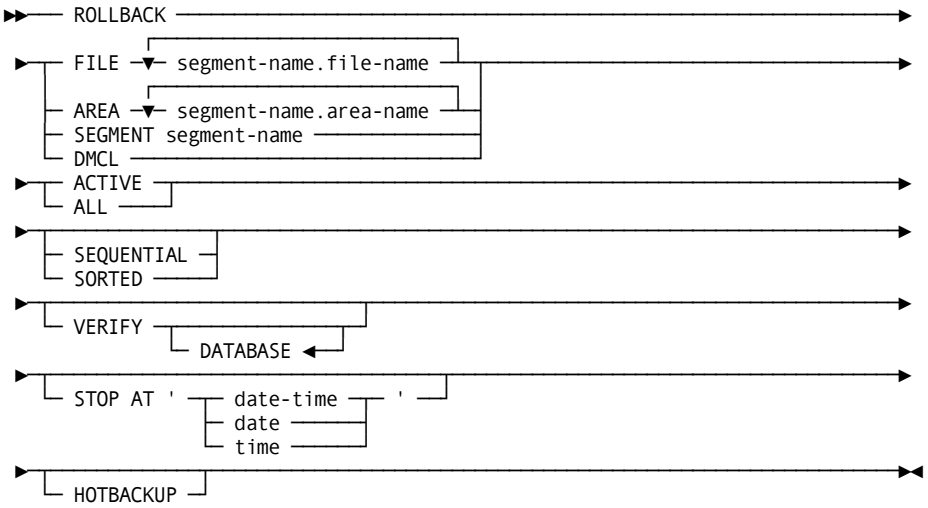
## Usage

*Verifying quiesce points:* A quiesce point is defined to be a point in time for which no recovery unit that is active has a before or after image on the journal file for the specified portion of the database. A recovery unit is represented by the journal images starting at a BGIN or COMT checkpoint and ending with a COMT, ABRT, or ENDJ checkpoint.

If a start time is specified, it must coincide with a quiesce point, otherwise the extract will terminate with an error. If no start time is specified, you can indicate whether or not the start of the journal file should be a quiesce point through the use of the VERIFY QUIESCE option. If specified, the extract will terminate with an error unless the start of the journal file coincides with a quiesce point.

## Rollback Utility Statement

## Syntax




## Parameters

VERIFY

VERIFY DATABASE

Verifies the after images in the journal file before applying the before images. If an after image in the journal file does not match the contents of the database, the roll back operation will terminate with an error.

By default, if you do not specify VERIFY, before images are copied back without first verifying the after images.

The optional DATABASE keyword is included for consistency with the ROLLFORWARD utility statement and has no impact on the processing of the VERIFY option.

HOTBACKUP

Indicates that the database has been restored from a backup file created with a hot backup procedure and that the journal images created while the backup was taking place are being rolled out.

### Usage

*When to use the HOTBACKUP parameter:* The HOTBACKUP parameter must be used when rolling out database updates made during the time that a hot backup was taking place. This parameter directs CA-IDMS to roll out changes made by aborted transactions in addition to those made by transactions that ended normally. If the database has not been restored from a hot backup, it is not necessary to roll out aborted transactions, because this was done by automatic recovery within the DC/UCF system. However, if the database was being copied at the time a transaction fails, a database page might have been copied to the backup file before automatic recovery had removed the effect of the failing transaction. For this reason, the HOTBACKUP parameter must be specified to ensure that the effect of failing transactions are removed from the database.

HOTBACKUP should only be specified when applying the before images created while the backup file was being created.

For more information on the use of the HOTBACKUP parameter and recovery from a hot backup, refer to *CA-IDMS Database Administration*.

## Unload/Reload/Fastload Extensions

The UNLOAD, RELOAD, and FASTLOAD utilities have been changed to support processing a database that contains duplicate record ids. This feature is necessary when unloading and reloading SQL-defined databases and user databases that contain duplicate record identifiers.

There are no special considerations in using this feature; however, new fields have been added to the record descriptor block that must be filled in by a user-written format program when executing the FASTLOAD utility. These new fields include an 18-byte record name and a 6-byte filler. Existing format programs must be changed to fill in these new fields regardless of whether or not the database contains duplicate record identifiers.

The format of the new record descriptor block follows. Fields one and two are new.

| Field | Usage | Size | Description |
|---|---|---|---|
| 1 | Char | 18 bytes | The record name of the record being loaded. The format program must initialize this field before calling IDMSDBLU. |
| 2 | Char | 6 bytes | Binary zeros |
| 3 | Binary | 4 bytes | The record id of the record being loaded. The format program must initialize this field before calling IDMSDBLU. |
| 4 | Binary | 4 bytes | The suggested page number for the record occurrence being loaded. Responsibility for supplying the page number depends on the location mode of the record:<br><br>If the location mode is either CALC or VIA a CALC record, IDMSDBLU determines and returns the suggested page number.<br><br>If the location mode is DIRECT, the format program must initialize this field either with an actual page number or with the value -1.<br><br>In the latter case, IDMSDBLU returns as the suggested page number the number of the first page in the range to which the record is assigned.<br><br>If the location mode is VIA a VIA record, the format program must process the owner record first and save the suggested page number of the owner record to initialize this field. |
| 5 | Binary | 4 bytes | Binary zeros |
| 6 | Binary | 4 bytes | Serial number that uniquely identifies the record occurrence being loaded.<br><br>IDMSDBLU generates and returns this value. |

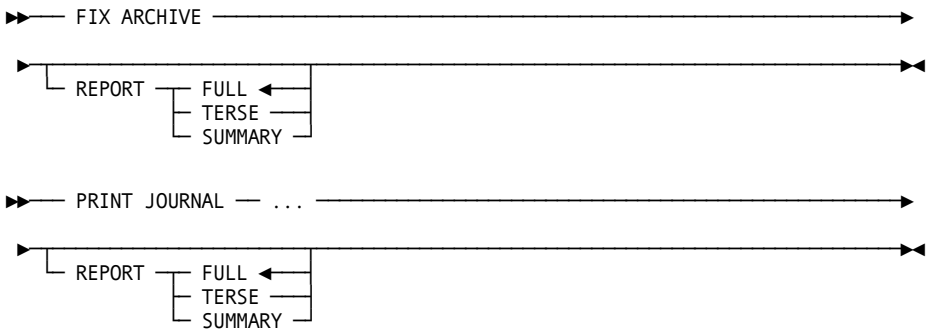
| Field | Usage | Size | Description |
|---|---|---|---|
| 7 | Char | 4 bytes | Error status. IDMSDBLU returns error-status codes that parallel those returned by CA-IDMS/DB. The format program must ensure that the returned value is 0000 before proceeding. |
| 8 | Char | Size of the largest record to be loaded | Actual record occurrence, left justified. The format program must initialize this field before calling IDMSDBLU. |

## Report Control for Print Journal/Fix Archive

The amount of output that is produced by the PRINT JOURNAL and FIX ARCHIVE utilities can now be controlled through a new option on the utility statement. This allows the output produced by these utilities to be tailored to the needs of a particular execution.

To use this feature, specify the REPORT option on the FIX ARCHIVE or PRINT JOURNAL utility statement.

### Syntax



### Parameters

REPORT

Specifies the amount of detail that is to appear on the output report.

FULL

Indicates that all details are to be reported. This includes for every transaction: checkpoints, database statistics, and area usage. In addition, transactions active at the end of the process are listed and the time of the last global quiesce point is identified. FULL is the default if no REPORT option is specified.

TERSE

Indicates that only transaction checkpoints and summary information is produced.

SUMMARY

Indicates that only final summary information is produced.

## ADS Compiler Enhancements

Two changes have been made to the ADS compilers to increase the integrity of the checkout process:

- Batch compilers now honor checkouts
- An optional APAR prevents the release of an entity with uncompiled changes

### Protection from Batch Compiles

ADSOBCOM and RHDCMAP1 will no longer permit structural updates to entities that are currently checked out. This prevents abnormal terminations of the online compilers caused because a change has been made to a checked-out entity by a batch compiler.

This protection is implemented through a new "check out" flag in the dictionary records that represent dialogs and maps (PROG-051 and MAP-098). The flag is set when an online compiler checks out one of these entities and is reset when the entity is released. The batch compilers (regardless of whether or not they are running in local mode or through CV), will test this flag before proceeding to make structural changes to the entity. If the flag is set, they will display an error message and proceed with the next map or dialog. If any compilation is terminated due to entity checkout, the condition code will be set to 8.

Protection from batch compiles is provided only for entities that are checked out under the Release 15.0 compilers.

### Preventing Release With Changes

An optional APAR is available with Release 15.0 that prevents the release of an entity by the online compilers if it has uncompiled changes.

Online compilers currently allow programmers to release maps, dialogs, and applications when their queues contain uncompiled changes. This allows a subsequent programmer to resume modification where the original programmer left off; however in practice, this often results in the accumulation of unassigned checkout queues and no record of who the original programmer was or what changes had been made.

The new optional APAR will cause the online compilers to display an error message rather than proceed with the release if uncompiled changes are found in the checkout queue.

As always, if no uncompiled changes are found, the release function will delete the checkout queue and a checkout queue with pending changes can still be reassigned by using the ADSM task.

The new behavior for the online compilers is established by turning on optional APAR bit number 212.

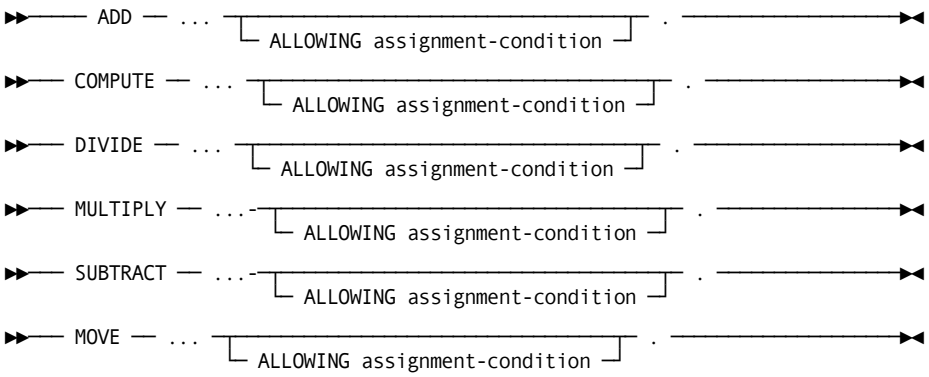
## Assignment Condition Handling

ADS now supports error handling for assignment and arithmetic commands. This allows an application to handle such errors as data exception or overflow rather than letting ADS abort dialog execution.

To use this feature, a new ALLOWING clause can be specified on any of the following commands: ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE, and MOVE. The ALLOWING clause specifies which error condition the dialog is prepared to handle. The possible conditions are listed below under "Assignment Command Status Condition." Specifying ANY-DATA-ERROR allows a dialog to test for any error condition.

Assignment command status conditions can also be used as conditional expressions in IF and WHILE commands in order to test the results of the previous assignment command.

## Syntax




## Parameters

ALLOWING *assignment-condition*

Specifies that if the named assignment condition occurs in processing the command, that control should be returned to the dialog rather than aborting the dialog. *Assignment-condition* must be one of the assignment command status conditions listed below.

## Example

The following example shows how the ALLOWING clause can be used to prevent application abends. The specified MOVE command moves a numeric field from an eight-byte field to a four-byte field. The application is prepared to handle any error condition that might arise.

```
move big-num to little-num allowing any-data-error.
if decimal-overflow-exception
    display error message text 'source data too large'.
if any-data-error
    display error message text 'invalid data value'.
```

### Assignment Command Status Condition

The following condition names can be specified as assignment command status conditions in ALLOWING clauses and conditional expressions.

```
ANY-DATA-ERROR
BAD-DATA-TYPE
UNSUPPORTED-DATA-CONVERSION
NO-NUMBER-EBCDIC/NUMERIC-CONVERSION
INCORRECT-FIELD-LENGTH
INVALID-SUBSCRIPT-VALUE
DATE-FORMAT-ERROR
SPECIFICATION-EXCEPTION
DATA-EXCEPTION
FIXED-POINT-OVERFLOW-EXCEPTION
FIXED-POINT-OVERFLOW-EXCEPTION
FIXED-POINT-DIVIDE-EXCEPTION
DECIMAL-OVERFLOW-EXCEPTION
DECIMAL-DIVIDE-EXCEPTION
FLOATING-POINT-DIVIDE-EXCEPTION
EXPONENT-OVERFLOW-EXCEPTION
EXPONENT-UNDERFLOW-EXCEPTION
SIGNIFICANCE-EXCEPTION
```

## OLQ DML User Exit

OLQ will now invoke a user exit prior to issuing any native DML command. This facility enables users to examine, modify, or disallow navigational access to data from within OLQ. This might be used for security enforcement, statistics gathering, or checking for special data values.

To use this feature, the OLQSDMLE program must be relinked to include IDMSBALI and the user-written exit program. The user-written program must have an entry point of OLQDMLX.

The exit program is called in user mode. The registers on entry are as follows:

- R1 — points to a parameter list that is the same as that generated for a native DML request. This is documented in the *CA-IDMS Programmer's Quick Reference Guide*.
- R13 — points to a 16-word save area in which the user exit should save the caller's registers.
- R14 — contains the address to which control should be returned.
- R15 — points to the OLQDMLX1 entry point within the user exit.

It is the user exit's responsibility to issue the DML command. If it decides that the command should not be executed, it must set the error status field in the IDMS communications block appropriately. The following is a sample program that can be used as a model for writing an OLQ DML exit.

## Sample Exit

```
OLQDMLX1 TITLE 'SAMPLE USER-WRITTEN DML EXIT FOR OLQ'
*OLQDMLX1 RENT EP=DMLXEP1 XA
************************************************************************
************************************************************************
**                                                                    **
**   THIS PROGRAM IS A TEMPLATE TO BE USED AS AN EXAMPLE FOR          **
**   PROVIDING ENTRY INTO AND EXIT FROM AN OLQ USER-WRITTEN           **
**   DBMS EDIT MODULE FOR RELEASE 14.0 AND LATER.                     **
**                                                                    **
************************************************************************
************************************************************************
**                                                                    **
**   THIS IS A SAMPLE ONLY AND NO GUARANTEE IS GIVEN AS TO            **
**   FUNCTIONALITY, ACCURACY, COMPLETENESS, OR PERFORMANCE.           **
**                                                                    **
************************************************************************
************************************************************************
         EJECT
*-----------------------------------------------------------------------
*
*  USERDMLX1 - USER-WRITTEN EXIT FOR DML COMMANDS IN OLQ
*
*-----------------------------------------------------------------------
*
*  OLQDMLX1 allows user-defined editing of DML commands before they
*  are issued by OLQ. The edit routine can be used for things such
*  as validating security, keeping statistics, looking for special
*  data values, etc.
*
*  If certain records, DML commands, or AREAs are to be selected
*  for editing, an IDMS database procedure should be used.
*
*  If many records or many subschemas are to be edited during OLQ
*  processing, this exit should be used.
*
*  OLQDMLX1 will be automatically called by OLQ before every
*  DML command if program OLQSDMLE is LINK/EDITed with this
*  module and with IDMSBALI and command 'DCMT VARY PROGRAM
*  OLQSDMLE NEW COPY' is issued.
*  //SYSLIB    DD DISP=SHR,DSN=IDMS.LOADLIB
*  //OBJLIB    DD DISP=SHR,DSN=USER.LOADLIB
*  //SYSLIN    DD *
*    INCLUDE SYSLIB(OLQSDMLE)
*    INCLUDE OBJLIB(OLQDMLX1)
*    INCLUDE SYSLIB(IDMSBALI)
*    MODE    AMODE(31),RMODE(ANY)
*    ENTRY   ENTRY
*    NAME    OLQSDMLE(R)
*
*   REGISTER USAGE -
*        R12 - BASE REGISTER
*        R13 -
*        R14 - RETURN ADDRESS FOR SUBROUTINES
*        R15 - A(DB/DC INTERFACE)
*        R0  -
*        R1  - A(PARAMETER LIST) AT ENTRY AND DURING CALLS
*        R2  - A(SECURITY REQUEST BLOCK)
*        R3  -
*        R4  -
*        R5  - WORK REGISTER
*        R6  -
*        R7  -
*        R8  - A(OLQ GLOBAL WORK AREA)
*        R9  -
```

```
*        R10 -
*        R11 -
*
*-------------------------------------------------------------------
         EJECT
DMLXSTG  DSECT
*-------------------------------------------------------------------
*        Any user-required storage is defined here
*-------------------------------------------------------------------
WORKAREA DC    CL80' '
DMLXSTGL EQU   *-DMLXSTG
SSCTRLDS DSECT
         @SSCTRL
EJECT
*-------------------------------------------------------------------
*        Entry code is defined here
*-------------------------------------------------------------------
         #MOPT CSECT=OLQDMLX1,ENV=USER
         @MODE MODE=IDMSDC,WORKREG=R0,QUOTES=YES,DEBUG=YES
         USING DMLXEP1,R12
         ENTRY DMLXEP1
         DC    0F'0',CL8'DMLXEP1'
DMLXEP1  DS    0F
         STM   R0,R15,0(R13)          SAVE OLQ'S REGISTERS
         LA    R13,16*4(,R13)         ADJUST STACK POINTER
         LR    R12,R15                ADDRESSIBILITY
         #GETSTG TYPE=(USER,SHORT),LEN=DMLXSTGL,ADDR=(R11),            X
               PLIST=*,STGID='USER',INIT=X'00'
         USING DMLXSTG,R11
         LR    R5,R13                 STACK POINTER
         SH    R5,=AL2(16*4)          A(OLQ'S REGISTERS)
         L     R14,56(,R5)            OLQ'S RETURN REGISTER
         CLC   =AL2(28),0(R14)        IF NOT A DML COMMAND
         BNE   DMLXEXIT                  CONTINUE DC PROCESSING
         L     R1,4(,R5)              RESTORE A(OLQ'S PARM LIST)
         L     R5,4(,R1)              A(SSCIDBCM+4)
         LA    R5,5(,R5)
         SR    R5,R1                  IDBMSCOM SUBSCRIPT
         EJECT
*-------------------------------------------------------------------
*        Edit code is defined here
*-------------------------------------------------------------------
*-------------------------------------------------------------------
*        'Bind Run Unit' edit code is defined here
*-------------------------------------------------------------------
DMLX1000 DS    0H
         CH    R5,=H'59'              IF NOT 'BIND RUN UNIT'
         BNE   DMLX2000                  SEE IF THIS IS 'OBTAIN'
*        Code 'BIND RUN UNIT' pre-processing here
         B     DMLXEXIT               PERFORM THE 'BIND'
         EJECT
*-------------------------------------------------------------------
*        DML edit code is defined here
*-------------------------------------------------------------------
DMLX2000 DS    0H
         CH    R5,=H'32'              IF NOT 'OBTAIN CALC'
         BNE   DMLX3000                  SEE IF THIS IS 'FINISH'
         L     R5,8(,R1)              A(RECORD NAME)
         CLC   =CL16'EMPLOYEE',0(R5)  IF NOT 'OBTAIN CALC EMPLOYEE'
         BNE   DMLXEXIT                  PERFORM THE 'OBTAIN CALC'
*        Code 'OBTAIN CALC EMPLOYEE' processing here
         L     R5,160(,R8)            A(RECORD IO BUFFER)
         CLC   =C'0048',0(R5)         IF EMP-ID-0415 NOT = 0048
         BNE   DMLXEXIT                  PERFORM THE 'OBTAIN CALC'
ABND2000 L     R5,0(,R1)              A(SSCTRL)
         USING SSCTRLDS,R5
```

```
         MVC   ERRSTAT,=C'0399'       'OBTAIN CALC EMPLOYEE' is
         MVC   ERRORREC,=C'SECURITY ERROR  '       not allowed
         B     DMLXRETN               RETURN TO OLQ
         DROP  R5
         EJECT
*-------------------------------------------------------------------
*        FINISH edit code is defined here
*-------------------------------------------------------------------
DMLX3000 DS    0H
         CH    R5,=H'2'               IF NOT 'FINISH RUN UNIT'
         BNE   DMLXEXIT                  PERFORM DML
*        Code 'FINISH RUN UNIT' processing here
         B     DMLXEXIT               PERFORM THE 'FINISH RUN UNIT'
         EJECT
*-------------------------------------------------------------------
*        Exit code is defined here
*-------------------------------------------------------------------
DMLXEXIT L     R15,=V(IDCSACON)       A(CSA)
         SH    R13,=H'64'             POINT TO OLQ'S STACK
         LM    R0,R14,0(R13)          RESTORE OLQ'S REGISTERS
         BR    R15                    EXECUTE REQUESTED COMMAND
DMLXRETN SH    R13,=H'64'             POINT TO OLQ'S STACK
         LM    R0,R15,0(R13)          RESTORE OLQ'S REGISTERS
         LA    R14,2(,R14)            A(NEXT INSTRUCTION)
         BR    R14                    RETURN TO OLQ
         DROP  R11
         LTORG
         END   DMLXEP1 x
```

## Assembly and Link Edit (OS/390)

```
//*-------------------------------------------------------------
//*            ASSEMBLER IEV90 JOB STREAM
//*-------------------------------------------------------------
//ASMSTEP EXEC PGM=IEV90,
//        PARM='ALIGN,XREF,PUNCH,NODECK',
//        REGION=2048K
//SYSLIB       DD  DSN=idms.maclib,DISP=SHR
//             DD  DSN=idms.srclib,DISP=SHR
//             DD  DSN=os390.maclib,DISP=SHR
//SYSUT1       DD  DSN=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT2       DD  DSN=&&SYSUT2,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT3       DD  DSN=&&SYSUT3,UNIT=VIO,SPACE=(1700,(600,100))
//SYSPRINT     DD  SYSOUT=*
//SYSPUNCH     DD  DSN=&&OBJECT,
                     DISP=(NEW,PASS),
                     UNIT=SYSDA,
                     SPACE=(80,(500,1000))
//SYSIN        DD  *
OLQ DML Exit program
//*-------------------------------------------------------------
//*            LINK IEWL
//*-------------------------------------------------------------
//LINK    EXEC PGM=IEWL,
//        PARM='LET,LIST,XREF,RENT',
//        REGION=128K,
//        COND=(8,LT,ASMSTEP)
//SYSLMOD      DD  DSN=idms.loadlib,DISP=SHR
//SYSPRINT     DD  SYSOUT=*
//SYSUT1       DD  DSN=&&SYSUT1,
//             UNIT=SYSDA,
//             SPACE=(6400,(80)),
//             DISP=(NEW,PASS)
//IN1          DD  DSN=idms.distload,DISP=SHR
//IN2          DD  DSN=&&OBJECT,DISP=(OLD,DELETE)
```

```
//SYSLIN       DD  DDNAME=SYSIN
//SYSIN        DD  *
 INCLUDE IN1(OLQSDMLE)
 INCLUDE IN2
 INCLUDE IN1(IDMSBALI)
 ENTRY ENTRY
 MODE AMODE(31),RMODE(ANY)
 NAME OLQSDMLE(R)
```

| Item | Description |
|---|---|
| idms.distload | data set name of the CA-IDMS SMP/E distribution load library |
| idms.loadlib | data set name of the CA-IDMS load library |
| idms.maclib | data set name of the CA-IDMS macro library |
| idms.srclib | data set name of the CA-IDMS source library |
| os390.maclib | data set name of the OS/390 system macro library |

### Assembly and Link Edit (VSE/ESA)

```
// DLBL   idmslib,
// EXTENT ,nnnnnn
// LIBDEF *,SEARCH=(idmslib.sublib)
// LIBDEF PHASE,CATALOG=idmslib.sublib
// OPTION CATAL
// EXEC ASMA90,SIZE=128K
OLQ DML Exit program
/*
  INCLUDE OLQSDMLE
  INCLUDE IDMSBALI
  ENTRY   ENTRY
// EXEC LNKEDT,SIZE=128K
/*
```

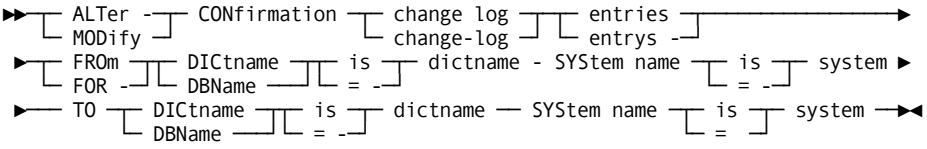
| Item | Description |
|---|---|
| idmslib | Filename of the file containing CA-IDMS modules |
| idmslib.sublib | Name of the sublibrary within the library containing CA-IDMS modules |
| Nnnnnn | Volume serial identifier of appropriate disk volume |

## CA-Endevor/DB Archive & Compress Enhancements

In Release 15.0, the Archive and Compress Utility, NDVRARCO, has been enhanced to allow the modification of dictionary identification information in Confirmation Change Log Entries (CLEs). This enables promotion of entities even if the name of the dictionary or the name of the system on which it resides, is changed.

To use this feature, execute the new ALTER CONFIRMATION command under the Archive and Compress utility.

### Syntax




### Parameters

`FROM DICTNAME`

Identifies the Confirmation Change Log Entries that are to be changed:

- *dictname*—Specifies the name of the dictionary whose entries are to be changed.
- *system*—Specifies the name of the system associated with the dictionary whose entries are to be changed. If the system name in the current Confirmation Change Log Entry is blank, specify *system* as '' (two single-quotes).

`TO DICTNAME`

Specifies the new dictionary identification information to which the Confirmation Change Log Entries will be changed:

- *dictname*—Specifies the dictionary name that will be set in the Confirmation Change Log Entry.
- *system*—Specifies the name of the system that will be set in the Confirmation Change Log Entry. To specify a blank system name, specify *system* as '' (two single-quotes).

### Usage

A maximum of 200 ALTER CONFIRMATION statements may be processed in a single Archive and Compress execution.

### Example

The following ALTER CONFIRMATION statement changes the system name in all Confirmation Change Log Entries with a dictionary name of 'APPLDICT' and a blank system name.

```
alter confirmation from dict appldict system ' ' to dict appldict system testsys
```

## SQL Procedures

Release 15.0 provides the ability to define and call a procedure using a simpler interface than that of table procedures. This new support is a subset of the SQL standard specification for external procedures and provides a way of invoking routines using a remote procedure call paradigm.

To use this feature, the following steps must be taken:

- Define the procedure using the new CREATE PROCEDURE statement.
- Write the procedure in COBOL, PLI, or Assembler following the guidelines outlined below. It may also be possible to use an existing program as a procedure.
- If necessary, define the program to a CA-IDMS system.
- Invoke the procedure from within a query-specification, a SELECT statement, or the new SQL CALL statement.

### Defining Procedures

SQL procedures are defined using the new CREATE PROCEDURE statement.

Similarly the new ALTER PROCEDURE and DROP PROCEDURE statements are used to modify and delete the definition of existing procedures.

Each of these statements is described in Appendix D, “New and Revised SQL Statements.”

## Writing Procedures

Refer to Appendix F, "Defining and Using Procedures" for a comprehensive discussion on writing procedures. The following considerations apply when writing the new procedures.

### Calls to the Procedure

When invoked, a procedure is called exactly once for each set of input values, regardless of the type of statement that contains the procedure reference. Within the single call, the procedure must use the input values, perform the expected action and return whatever output values are appropriate. This differs from a table procedure which may be called several times for a given set of input values depending on the type of statement containing the procedure reference.

### Calling Arguments

The calling arguments that are passed to a procedure are the same as those passed to a table procedure, but not all arguments need to be processed by the procedure. Refer to Appendix F, "Defining and Using Procedures" for a complete description of the calling arguments passed to a procedure.

When a procedure is invoked, the SQL Command Code is always 16 (FETCH), the SQL Operation Code is always 16 (Next Row) and the Instance Identifier is always a unique value since there is no concept of "scan" for non-table procedures. The values in the remaining arguments and their use are the same as those for a table procedure.

In simple cases, i.e., when all parameters are not null or defined with default and no work areas need to be allocated, the program called by a non-table procedure only needs to define and act upon the parameters specified in the procedure definition; null indicator parameters and common parameters for communication with CA-IDMS are optional.

### Local Work Area

Because there is no concept of calls related through a scan for non-table procedures, each call to such a procedure receives a newly-allocated local work area. Consequently, passing information from one call to another must be done through a global rather than a local work area.

### SQLSTATE Values

A procedure can signal exceptional conditions by setting the SQLSTATE value appropriately; that is, an SQLSTATE value of 02000 indicates that no data will be returned. The initial value of SQLSTATE is 00000.

### Referencing Procedures

Unlike table procedures, which can be referenced anywhere that a table can be referenced, non-table procedures can be referenced in only one of three locations:

- In the FROM clause of a query-specification
- In the FROM clause of a SELECT statement
- In a CALL statement

Refer to Appendix G, "Sample COBOL Procedure" for an example of how to define, code, and invoke a procedure.

## SQL Call Statement

Release 15.0 supports the SQL CALL statement which allows procedures to be invoked using a remote procedure call paradigm. This statement is part of the current SQL standard specification.

If the CALL statement is used to invoke a table procedure, the procedure will be invoked as if it was referenced in a SELECT statement, meaning that it will be invoked once for Open Scan, one or more times for Next Row and once for Close Scan. If it is used to invoke a procedure, the procedure will be called only once.

The following is an example of a CALL statement:

```
call newhire ('FRED', 'SMITH', '1954-06-23', 123444555)
```

When invoked through the Command Facility, the CALL statement will result in a set of output values for each parameter defined to the procedure.

The CALL statement is not supported in any current version of CA-IDMS Server nor Enterprise Gateway.

For a complete description of the CALL statement, refer to Appendix D, "New and Revised SQL Commands."

## Extended Use of Subquery

Release 15.0 enables an SQL user to use a subquery as the source value for a column in an UPDATE statement. This extends the power of the update capability in SQL and is another step toward compliance with the current SQL standard.

The following is an example of the extended use of a subquery in an UPDATE statement:

```
update department d set salary_budget =
   (select 1.1 * sum (salary) from employee e
      where e.deptid = d.deptid)
```

The enhanced UPDATE statement syntax is described in Appendix D, "New and Revised SQL Statements."

## DMLO Enhancements

DMLO now provides the ability for a site to specify the default DBNAME that is to be used whenever DMLO is invoked. To do this, the USDTPARM module must be recompiled specifying a value for the following parameter:

DEFDBNM — the default DBNAME used on the DMLO signon screen

If no value is specified for the parameter, it defaults to blanks as in prior releases.

To recompile and relink USDTPARM module, use the sample job UMOD1 in the SAMPCL library.

# Performance Enhancements

## UAB Storage Management

The management of User Attribute Blocks (UABs) has been altered in release 15.0 so that it is more efficient. Additionally the size of a typical UAB has been reduced. These changes result in less storage fragmentation, reduced paging and lower CPU usage.

No action is necessary to use this feature.

## Lock Manager Multitasking

The management of transaction locks has been altered in release 15.0 to increase concurrency in a multi-tasking environment. The lock manager now single-threads at an individual resource level rather than at a hash entry level. In a multi-tasking environment, this change results in higher throughput and reduced CPU usage.

No action is necessary to use this feature.

Appendix

A

# New and Revised DCMT Commands

## DCMT DISPLAY AREA

The output of this command now shows the following information:

- The sharability state of the area
- Whether or not there is inter-CV-interest in the area
- Whether or not the area status has been established with the PERMANENT option
- If the area is the target of an outstanding quiesce operation, whether it is quiescing or quiesced
- If the area is the target of an outstanding vary operation, the status to which it is being varied

### Example

The output from the following command is shown illustrating the display for a non-shared area whose status was assigned with the PERMANENT option.

**dcmt display area empdemo.emp-demo-region**

```
 D AREA EMPDEMO.EMP-DEMO-REGION
---------- Area -----------   Lock    Lo-Page    Hi-Page #Ret  #Upd #Tret #Ntfy
EMPDEMO.EMP-DEMO-REGION        Upd      75001      75100    0     0     0     0
 Stamp: 1998-11-17-09.55.31.875826  Pg grp: 0      NoShare  NoICVI  Perm
```

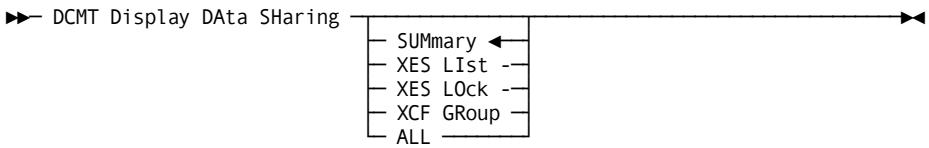
The following output for the same command shows the area in a quiescing state.

```
D AREA EMPDEMO.EMP-DEMO-REGION
---------- Area -----------   Lock    Lo-Page    Hi-Page #Ret  #Upd #Tret #Ntfy
EMPDEMO.EMP-DEMO-REGION        Upd      75001      75100    0     0     0     0
 Stamp: 1998-11-17-09.55.31.875826  Pg grp: 0      NoShare  NoICVI  Perm
           *** area is QUIESCING ***
```

# DCMT DISPLAY DATA SHARING

This new command displays information about the data sharing environment.

## Syntax




## Parameters

`SUMmary`

Displays summary information about this system's data sharing group. SUMmary is the default if no option is specified.

`XES LIst`

Displays information about the coupling facility list structure associated with this system's data sharing group.

`XES LOck`

Displays information about the coupling facility lock structure associated with this system's data sharing group.

`XCF GRoup`

Displays information about the members of this system's data sharing group and messages that have been sent between those members.

`ALL`

Displays information about the members, list and lock structures associated with this system's data sharing group. It includes all information displayed for each of the above options.

## Examples

### dcmt display data sharing summary

```
DISPLAY DATA SHARING SUMMARY
***  Display Data Sharing request  ***
Group name                DBDCGRP1
Default Cache             NULL
On Connectivity Loss      NOABEND
```

```
Group member SYSTEM73 is Active
   Prior CV state: Ready                Current CV state: Active
   LmgrProxy recovery locks          0  LmgrResource recov. locks          0
Group member SYSTEM74 is Active
   Prior CV state: Ready                Current CV state: Active
   LmgrProxy recovery locks          0  LmgrResource recov. locks          0

Structure CAIDMSDBDCGRP1LI type LIST
   Actual size (K)                4096  Connect id         2

Structure CAIDMSDBDCGRP1LK type LOCK
   Actual size (K)                5120  Connect id         2
   Lock entries                  65536  Max. connections  32
Record Data Entry information:
   Maximum number                33385  Nr of times SOS                    0
   Currently in use                769  Held by this CV                  214
   HWM                             771  Freeable by this CV              111
```

## dcmt display data sharing xes list

```
DISPLAY DATA SHARING XES LIST
***  Display Data Sharing request  ***
Structure CAIDMSDBDCGRP1LI type LIST
   Actual size (K)                4096  Connect id         2

List structure statistics
List name    * Reads      * Writes     * Deletes    * VersionErr * Errors
List 0       *          0 *          0 *          0 *          0 *          0
AreaList     *         15 *          8 *          0 *          0 *          0
FileList     *         41 *         34 *          0 *          0 *          0
QueueList    *          0 *          0 *          0 *          0 *          0
List 4       *          0 *          0 *          0 *          0 *          0
List 5       *          0 *          0 *          0 *          0 *          0
```

## dcmt display data sharing xes lock

```
DISPLAY DATA SHARING XES LOCK
***  Display Data Sharing request  ***
Structure CAIDMSDBDCGRP1LK type LOCK
   Actual size (K)                5120  Connect id         2
   Lock entries                  65536  Max. connections  32
Record Data Entry information:
   Maximum number                33385  Nr of times SOS                    0
   Currently in use                769  Held by this CV                  214
   HWM                             771  Freeable by this CV              111

Lock structure statistics (Obtain)
ResType      * Obtains    * Obt.Async  * Obt.Syncf  * Obt.Denied * Obt.Except
LmgrResource *        894 *         67 *          0 *          0 *          6
Phys.Page    *       1021 *        303 *          0 *          0 *          0
GlobalDeadLk *         73 *          0 *          0 *          0 *          0
LmgrProxy    *       1320 *        205 *          0 *          0 *          0
EnqDeq       *          0 *          0 *          0 *          0 *          0
AreaList     *          3 *          0 *          0 *          0 *          0
FileList     *         26 *          9 *          0 *          0 *          0
GlobalQueue  *          0 *          0 *          0 *          0 *          0

Lock structure statistics (Alter, Release)
ResType      * Alters     * Alt.Async  * Releases   * Rel.Async  * Alt+Rel.Exc
LmgrResource *        949 *        445 *        894 *         89 *         71
```

```
Phys.Page    *        848 *         41 *        557 *        317 *          0
GlobalDeadLk *          0 *          0 *         73 *          0 *          0
LmgrProxy    *       1067 *        311 *        942 *        118 *         72
EnqDeq       *          0 *          0 *          0 *          0 *          0
AreaList     *          0 *          0 *          3 *          0 *          0
FileList     *          0 *          0 *         26 *         10 *          0
GlobalQueue  *          0 *          0 *          0 *          0 *          0

Lock structure statistics (Miscellaneous)
ResType      * ContExit   * NotifExit
LmgrResource *        354 *        251
Phys.Page    *        981 *        318
GlobalDeadLk *          0 *          0
LmgrProxy    *        880 *        310
EnqDeq       *          0 *          0
AreaList     *          0 *          0
FileList     *         22 *          0
GlobalQueue  *          0 *          0
```

## dcmt display data sharing xcf group

```
DISPLAY DATA SHARING XCF GROUP
***  Display Data Sharing request  ***
Group name               DBDCGRP1
Default Cache            NULL

Group member SYSTEM73 is Active
   Prior CV state: Ready                 Current CV state: Active
   LmgrProxy recovery locks          0   LmgrResource recov. locks          0
XCF Message statistics:
MessageType  * Sends      * SendErrors * Receives   * RecvPurged * RecvErrors
Reply        *          0 *          0 *          0 *          0 *          0
TestMsg      *          0 *          0 *          0 *          0 *          0
SyncStamp    *          0 *          0 *          0 *          0 *          0
GlobalDeadLk *          0 *          0 *          0 *          0 *          0
DCMTDCUFSEND *          0 *          0 *          0 *          0 *          0
AreaFileVal  *          0 *          0 *          0 *          0 *          0
QueueMsg     *          0 *          0 *          0 *          0 *          0
ProgramMsg   *          0 *          0 *          0 *          0 *          0

Group member SYSTEM74 is Active
   Prior CV state: Ready                 Current CV state: Active
   LmgrProxy recovery locks          0   LmgrResource recov. locks          0
XCF Message statistics:
MessageType  * Sends      * SendErrors * Receives   * RecvPurged * RecvErrors
Reply        *        172 *          0 *        178 *          0 *          0
TestMsg      *          0 *          0 *          0 *          0 *          0
SyncStamp    *          0 *          0 *          0 *          0 *          0
GlobalDeadLk *        318 *          0 *        270 *          0 *          0
DCMTDCUFSEND *          6 *          0 *          4 *          0 *          0
AreaFileVal  *          0 *          0 *          2 *          0 *          0
QueueMsg     *          0 *          0 *          0 *          0 *          0
ProgramMsg   *          0 *          0 *          0 *          0 *          0
```

## Usage

*Display for DCMT DISPLAY DATA SHARING SUMMARY*: The following information is displayed for the SUMMARY option:

- The name of this system's data sharing group, the name of its default cache, and its connectivity loss setting.
- A list of the members of the group showing:
  - Their member name
  - Their member state as assigned by XCF
  - Their prior and current user states as assigned by IDMS
  - The number of recovery locks held on proxies and resources (records and areas) on behalf of the member if it requires recovery
- The name of the list structure associated with this system's data sharing group, its size and this system's list structure connection identifier.
- The following information about the lock structure associated with this system's data sharing group:
  - Its name
  - Its size
  - This system's lock structure connection identifier
  - The number of lock entries in the structure
  - The maximum number of CA-IDMS systems that can be members of the group
  - The maximum number of record data entries that can be contained in the lock structure
  - The number of times the lock structure has run short on record data entries
  - The number of record data entries currently allocated
  - The number allocated by this system
  - The highest number of record data entries allocated at one time
  - The number of record data entries that are held by this member and that are freeable because they are held on behalf of unused proxies

*Display for DCMT DISPLAY DATA SHARING XES LIST:* The following information is displayed for the XES LIST option:

- The name of the list structure associated with this system's data sharing group, its size and this system's list structure connection identifier.
- A set of statistics associated with each list in the list structure showing for each:

- Its name or list identifier. AreaList maintains information about shared areas. FileList maintains information about files associated with shared areas. QueueList maintains information about shared queues.
- The number of reads issued for entries in the list.
- The number of writes issued for entries in the list.
- The number of deletes issued for entries in the list.
- The number of conflicts detected when updating a list entry.
- The number of other errors detected when accessing the list.

*Display for DCMT DISPLAY DATA SHARING XES LOCK:* The following information is displayed for the XES LOCK option:

- The summary information about the lock structure as described above for the SUMMARY option.
- A set of statistics by global resource type. For each type of resource controlled through the lock structure, the display shows:
  - The resource type. LmgrResource represents a record or area. Phys.Page represents a database page. GlobalDeadLk represents a resource used to single thread assignment of a global deadlock detector. LmgrProxy represents a proxy. EnqDeq represents an enqueued resource. AreaList represents a resource used to single thread update of the coupling facility area list. FileList represents a resource used to single thread update of the coupling facility file list. GlobalQueue represents a shared queue.
  - The number of lock acquisition requests (obtains) that were issued by this system.
  - The number of obtains not serviced immediately.
  - The number of times an obtain failed because a wait was required and a wait was not allowed.
  - The number of times an obtain was denied by the CA-IDMS contention exit.
  - The number of times other exception conditions were encountered on an obtain request.
  - The number of times a request was issued to alter the state of a global lock held by this system.
  - The number of times an alter could not be serviced immediately.
  - The number of lock release requests issued by this system.
  - The number of times a lock release could not be serviced immediately.
  - The number of times an exception condition was encountered on an alter or release request.

  - The number of times the CA-IDMS contention exit was invoked to resolve conflicts for the resource type.
  - The number of times the CA-IDMS notify exit was invoked as part of conflict resolution.

*Display for DCMT DISPLAY DATA XCF GROUP:* The following information is displayed for the XCF GROUP option:

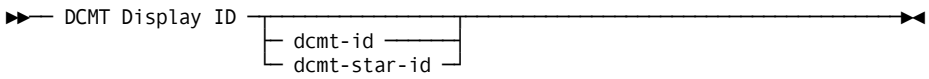
- The name of this system's data sharing group and the name of this system's default cache.
- A list of the members of the group showing:
  - Their member name
  - Their member state as assigned by XCF
  - Their prior and current user states as assigned by IDMS
  - The number of recovery locks held on proxies and resources (records and areas) on behalf of this member if it requires recovery
- A set of statistics for each member showing by message type the following information:
  - The type of message. Reply represents replies issued by this system to messages sent by the indicated member. SyncStamp messages inform members of changes in SQL table definitions. GlobalDeadLock messages are used to resolve global deadlocks. DCMTDCUFSEND messages are used to broadcast commands. AreaFileVal messages inform members of changes in area and file status. QueueMsg messages inform members of the creation of a new global queue. ProgramMsg messages inform members whenever a #DELETE … NEW COPY is issued.
  - The number of messages sent to the member from this system.
  - The number of errors encountered when sending a message to the member.
  - The number of messages received from the member by this system.
  - The number of messages sent to this system by the member that were purged because no task was registered to receive that type of message.
  - The number of errors encountered in attempting to receive a message from the member.

For information on the meaning of these parameters, refer to the topic Monitoring Data Sharing Groups in Chapter 3,"Exploiting the Parallel Sysplex Architecture."

# DCMT DISPLAY ID

This new command displays outstanding DCMT operations.

## Syntax




## Parameters

ID

Identifies the DCMT operations to be displayed.

- If no identifier is specified, the status of all outstanding DCMT operations will be displayed.
- *dcmt-id* – Specifies the identifier of the DCMT operation to be displayed.
- *dcmt-star-id* – Specifies that all dcmt operations whose identifier begins with the specified alphanumeric characters be displayed. Dcmt-star-id is a character string whose last character is an asterisk (*) that denotes a wild card character.

  In this example, CA-IDMS will display all DCMT operations whose identifier begins with CUST:

  dcmt d id cust*

## Usage

*Outstanding DCMT operations:* The following DCMT commands can be assigned identifiers. These are the only DCMT operations that will appear in the DCMT DISPLAY ID output.

- DCMT VARY AREA
- DCMT VARY SEGMENT
- DCMT QUIESCE AREA/SEGMENT/DBNAME

*Output:* The following information will be displayed for a DCMT operation:

- The nodename on which the DCMT command executed. In a data sharing environment, this is the same as the member name of the originating node. In a non-data sharing environment this is the nodename of the current node.
- The DCMT identifier assigned to the operation
- A description of the operation

- The status of the operation.

  The following table describes the possible status values for a VARY area operation.

| Status | Meaning |
|---|---|
| Initiating | The operation is initiating |
| Stop Upd | The vary operation is executing. No new update accesses are allowed to the area. |
| Stop Ret | The vary operation is executing. No new retrieval or update accesses are allowed to the area. |
| Deferred | The vary operation is waiting for conflicting tasks and user sessions to end |
| Processed | The vary operation has completed successfully and is terminating. |
| Terminating | The vary operation is terminating due to being cancelled by a DCMT VARY ID command. |

The following table describes the possible status values for a QUIESCE operation.

| Status | Meaning |
|---|---|
| Unknown | The operation is initiating. |
| Quiesce aborted | The quiesce operation is in the process of terminating. |
| Quiesce ended | The quiesce operation has completed and is terminating. |
| Quiesced | The quiesce point has been reached. |
| Quiesced (Locl) | In a data sharing environment, the target areas have been quiesced locally. |
| Quiesced (Gbl) | In a data sharing environment, the target areas have been quiesced globally. This status appears only on the originating node. |
| Quiescing | The target areas are being quiesced. |
| Quiescing (Locl) | In a data sharing environment, the target areas are being quiesced locally. |
| Quiescing (Gbl) | In a data sharing environment, the target areas are being quiesced globally. This status appears only on the originating node. |

## Example

The following is an example of the output from a DCMT DISPLAY ID command.

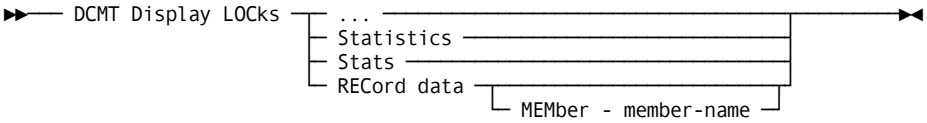
**dcmt display id**

```
D ID
-Origin-  -- ID --  ----------------- Command -----------------  --- Status ----
SYSTEM74  EMPBKP    QUIESCE SEGMENT EMPDEMO                     Quiesced
```

# DCMT DISPLAY LOCK

This command has been revised to display statistical information about transaction lock management.

## Syntax




## Parameters

`Statistics/Stats`

Requests the display of statistical information associated with management of transaction locks.

`RECord data`

Requests the display of record data entries held by a member of a data sharing group.

`MEMber member-name`

Specifies the name of the member for which record data entries are to be displayed. If member-name is omitted, the record data entries held by the member on which the command is executed are displayed.

## Example

The following example shows the output from the following DISPLAY LOCK STATS command:

### dcmt display lock stats

```
DISPLAY LOCK STATS
                      *** Transaction Lock Statistics ***
                   Local Trans      Local Page    Global Proxy Global Resource
Lock Requests           101393            7666           15567           14237
Locks Held                  12             666            1085               0
Rec Data Held                                               802               0
Waits                        0                             5245            5869
Locks Denied                46                                0              35
New Contention                                             1026             403
Contention Xit                                             3905            1463
Notify Xit                                                  850             650
  Downgrades                                                103             306
  Releases                                                   86             101
  Upgrade Posts                                             340

                    - - - -  Notify/Longterm Stats - - - -
                             Notify        Longterm Share       Longterm Update
Acquired                      36129                     0                     0
Held                              0                     0                     0

Global Notifies                Out                    In
          Proxy                  0                     0
       Resource                 48                    40

                    - - - - - -  Proxy Management  - - - - -
                        Created           Freed          Reused          Stolen
                           2748            1743             501            7141

                    - - - - - Storage Management - - - - -
SYSLOCKs value:             5000
                  # Times Ovfl   # Ovfl Getstg  Curr Ovfl Size   Ovfl Size HWM
Overall:                     1              13          189184          189184
   Session:                  0               0               0               0
     Class:                  0               0               0               0
  Resource:                  1               7          136192          136192
  XES Reqs:                  0               0               0               0
     Proxy:                  1               6           52992           52992

                    - - - - - -  Miscellaneous - - - - - -
Upgrade Reqs:         31405    In Place:       26481     Denied:           35
Cleanup Calls:            0    Compression Calls:                   7
```

The following example shows the output from the DISPLAY LOCK RECORD DATA command:

### dcmt display lock record data

```
DISPLAY RECORD DATA
Record data entries for: SYSTEM74
ID     Version     Type            Resid          Mode    Ldel     Task
02     0020CD5       P    00000008 000124FF        X+               99
02     0020CD5       P    000F0008 000A8E4F        X+              144
02     0020CD5       P    000F0008 000A7103        X+              144
```

## Usage

*Display for DCMT DISPLAY LOCK STATISTICS*: The following information is displayed if the LOCK STATISTICS option is specified:

- For each of local transaction locks, local page locks, global proxy locks and global resource locks, the following information:
  - The number of lock requests issued
  - The number of locks currently held
- For each of local transaction locks, global proxy locks and global resource locks, the following information:
  - The number of record data entries held for global locks
  - The number of times a task waited on a lock request
  - The number of locks denied due to conflicts
- For global proxy and resource locks, the following information:
  - The number of times the CA-IDMS contention exit was invoked for a new contention situation
  - The number of times the CA-IDMS contention exit was invoked to resolve contention
  - The number of times the CA-IDMS notify exit was invoked to help resolve contention or to inform of DBMS activity for a record on which this system holds a notify lock
  - The number of times the CA-IDMS notify exit downgraded a global lock in an effort to eliminate contention
  - The number of times the CA-IDMS notify exit released a global lock in an effort to eliminate contention
  - The number of times the CA-IDMS notify exit upgraded local locks on resources represented by a proxy to global locks in order to provide a finer level of contention management
- The number of notify, longterm exclusive and longterm share locks that have been acquired and that are currently held.
- The number of cross-member notifications of DBMS activity that were issued by this system and that were received by this system as a result of notify locks placed on proxies and dbkeys.
- The number of proxy control blocks that were created, released, reused for the same proxy before being released, stolen from another proxy for which no lock was held.
- The value of the SYSLOCKs sysgen parameter. This parameter influences the amount of storage initially allocated for a number of the lock-related control blocks.

- Information on storage overflows for each of the following types of control blocks: session, lock class, resource, XES lock request block, and proxy. The following statistics are displayed for each:
  - The number of times a new overflow situation occurred
  - The number of times storage was acquired to increase the available number of control blocks
  - The current amount of overflow storage in use
  - The maximum amount of overflow storage at any one time.
- The number of times a longterm or notify lock was upgraded to a new mode and of those, the number that occurred without internally acquiring a new lock and the number that were denied due to a deadlock situation.
- The number of times the lock manager scanned all outstanding locks in order to locate and release those for a failing task.
- The number of times the lock manager eliminated duplicate kept locks for a task.

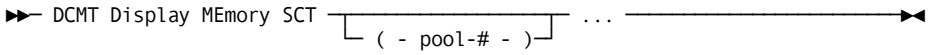
*Display for DCMT DISPLAY LOCK RECORD DATA:* The following information is displayed if the RECORD DATA option is specified:

- For each record data entry held on behalf of the specified member, the following information:
  - The lock structure connect id assigned to the member by the operating system
  - The version of the lock structure connection assigned by the operating system
  - The type of resource represented by the record data entry: a "P" indicates a proxy, an "R" indicates an area or dbkey
  - The resource identifier of the resource represented by the record data entry
  - The lock mode held by the member on the resource
  - A logical deletion indication. A "Y" in this field indicates that the record data entry is logically deleted
  - The IDMS/DC task identifier that resulted in the record data entry being created

# DCMT DISPLAY MEMORY

This command has been revised to allow the specification of a storage pool number when displaying an SCT.

## Syntax




## Parameters

*pool-#*

Specifies the storage pool number for which the storage control table (SCT) is to be displayed.

# DCMT DISPLAY MT

This new command displays the multitasking queue depth in effect for the system.

## Syntax


## Example

The following example shows the output from the following command:

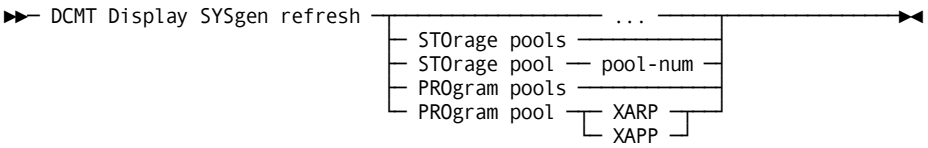
**dcmt display mt queue depth**

```
DCMT DISPLAY MT QUEUE DEPTH
MT Queue Depth is 002
```

# DCMT DISPLAY SYSGEN REFRESH

This command has been revised to display the pending changes for storage and program pools.

## Syntax




## Parameters

`STOrage pools`

Specifies that dynamic sysgen changes for all XA storage pools should be displayed.

`STOrage pool` *`pool-num`*

Specifies that dynamic sysgen changes for the specified storage pool should be displayed.

`pool-num`

Identifies the number of the XA storage pool for which dynamic sysgen changes should be displayed.

`PROgram pools`

Specifies that dynamic sysgen changes for all program pools should be displayed.

`PROgram pool XARP/XAPP`

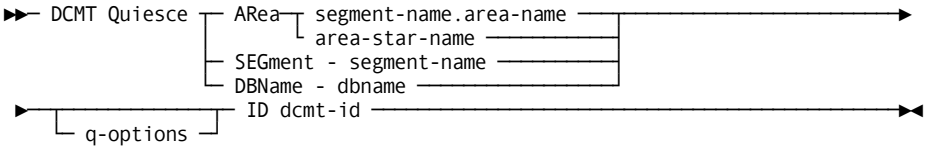
Specifies that sysgen changes for the specified program pool should be applied.

- XARP—Indicates that sysgen changes for the XA reentrant program pool should be applied.
- XAPP—Indicates that sysgen changes for the XA non-reentrant program pool should be applied.

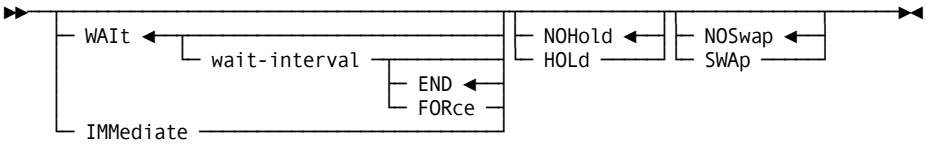
# DCMT QUIESCE AREA

This new command initiates a quiesce operation for one or more target areas.

## Syntax




Expansion of **q-options**




## Parameters

ARea

Indicates that one or more areas are to be quiesced. Valid values are:

- *segment-name.area-name* – Specifies the name of the area to be quiesced.
- *area-star-name* – Specifies that all areas whose name begins with the specified alphanumeric characters will be quiesced. Area-star-name is a character string whose last character is an asterisk (*) that denotes a wild card character.

  In this example, CA-IDMS will quiesce all areas whose segment name begins with PROD:

  dcmt q area prod*

SEGment *segment-name*

Specifies that all areas associated with the named segment are to be quiesced.

DBName *dbname*

Specifies that all areas associated with segments that are included in the named dbname are to be quiesced.

q-options

Specifies the options that are to be used for this quiesce operation.

`WAIt`

Specifies that the quiesce operation will wait for conflicting tasks or user sessions to relinquish update control of the area. This is the default behavior if neither WAIT nor IMMEDIATE is specified.

*`wait-interval`*

Specifies the amount of time the quiesce operation will wait for conflicting tasks or user sessions to relinquish update control of the area. If wait interval is not specified, the quiesce operation will wait indefinitely.

`END`

Specifies that if the areas cannot be quiesced within the specified wait interval, the quiesce operation will terminate. This is the default behavior if neither END nor FORCE is specified.

`FORce`

Specifies that if the areas cannot be quiesced within the specified wait interval, conflicting tasks and user sessions will be cancelled in order to reach a quiesce point.

`IMMediate`

Specifies that the quiesce operation will immediately cancel any tasks or user sessions that are accessing a target area in an update mode.

`NOHold`

Specifies that once the quiesce point has been established, the quiesce operation will automatically terminate. This is the default if neither HOLD nor NOHOLD is specified.

`HOLd`

Specifies that once the quiesce point has been established, the quiesce operation will continue until explicitly terminated by a DCMT VARY ID command.

`NOSwap`

Specifies that no journal swap should be initiated automatically once the quiesce point has been established. This is the default if neither SWAP nor NOSWAP is specified.

`SWAp`

Specifies that once the quiesce point has been established, a journal swap will be initiated.

*dcmt-id*

Specifies the identifier that is to be assigned to this quiesce operation. *Dcmt-id* must be a 1 – 8 alphanumeric character string that is unique across all outstanding DCMT operations originating on this node.

The identifier can subsequently be used to monitor or terminate the quiesce operation using DCMT DISPLAY ID and DCMT VARY ID commands.

## Usage

*Broadcasting quiesce commands*: There is no need to broadcast a quiesce command to more than one member of a data sharing group in order to quiesce shared areas. Subject to the scoping rules discussed below, shared areas will automatically be quiesced across all group members. Broadcasting a quiesce command to more than one member is effectively the same as issuing the command independently on each of those members. In neither case are the quiesce points for the separate quiesce operations synchronized.

*Scope of quiesce within a data sharing group* – In a data sharing environment, only areas that are accessible through the member on which the command originated will be quiesced. For example, consider a data sharing group of two members: CV1 and CV2 and the following command:

**dcmt quiesce area emp***

Member CV1 has one matching segment: EMPEAST; while member CV2 has two matching segments: EMPEAST and EMPWEST. If the above command is issued on CV1, only segment EMPEAST will be quiesced. If the same command is issued on CV2, both EMPEAST and EMPWEST will be quiesced.

Furthermore, only areas that are designated as shared in the member on which the command originates will be quiesced across all members of the group. Non-shared areas will be quiesced only within the member on which the command is issued. Therefore, in order to quiesce an area that is being updated by more than one member, the quiesce command must be issued on a member in which the area is designated as shared. The status of the area is not important. Even an area whose status is offline will be quiesced across all members of the group if it is designated as shared.

*Forcing a quiesce point:* A quiesce point can be forced either by specifying the IMMEDIATE option or by specifying a wait interval with the FORCE option. In either case, in order to achieve a quiesce point, CA-IDMS will:

- Cancel all tasks that are accessing a target area in an update mode
- Terminate all user sessions with no active task if they hold an update lock on a target area (by performing the equivalent of a DCMT VARY LTERM … RESOURCES DELETE)

- Vary offline all predefined system run units that are accessing a target area in an update mode (by performing the equivalent of a DCMT VARY RUNUNIT … OFFLINE)

If predefined system run units are varied offline in order to achieve the quiesce point, they will be varied online when the quiesce operation terminates.

*DC/UCF system termination:* If a quiesce operation is in progress and the DC/UCF system on which it was initiated is shutdown or abnormally terminates, the quiesce operation terminates also.

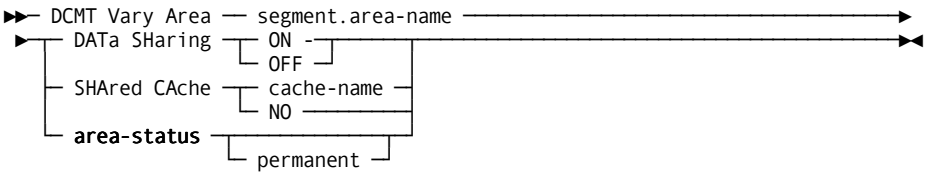
In a data sharing environment, if a member which is participating in a quiesce operation that was initiated on another system is shutdown or abnormally terminates, the quiesce operation continues. If the participating member is terminated in an orderly fashion using a DCMT SHUTDOWN command (with or without the IMMEDIATE option), its shared areas are quiesced as part of the shutdown operation. If the participating member abnormally terminates before it had locally quiesced the shared areas, the quiesce operation will be unable to complete until the participating member is restarted.

# DCMT VARY AREA

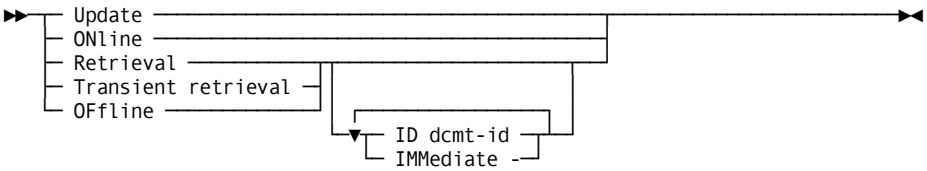
This command has been revised in the following ways:

- The sharability state of an area can now be changed
- The semantics associated with the shared cache parameter have been revised slightly and "AVAILABLE" has been removed as an option
- A change in area status can now be retained across system shutdowns
- An option has been added to specify that conflicting tasks should be cancelled if they prevent the area from being varied
- Certain vary area commands can be cancelled by means of the DCMT VARY ID command.

## Syntax




Expansion of **area-status**




## Parameters

`DATa SHaring`

Specifies the sharability state of the named area. The change will be made only if the area status is OFFLINE. Valid values are:

- ON—Specifies that this system is eligible to share update access to the area with other members of the system's data sharing group.
- OFf—Specifies that this system will no longer be eligible to share update access to the area with other members of the system's data sharing group.

`SHAred CAche`

Specifies the name or status of shared cache for all files associated with the named area. Valid values are:

- *cache-name*—Specifies that all files associated with the named area are to be assigned to the named cache structure. Cache-name must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.
- NO—Specifies that the files associated with the named area are no longer assigned to a cache structure.

`area-status`

Specifies the status that is to be assigned to the area. Refer to the *CA-IDMS System Tasks and Operator Commands* for a description of the valid values.

`dcmt-id`

Specifies the identifier that is to be assigned to this vary operation. *Dcmt-id* must be a 1 – 8 alphanumeric character string that is unique across all outstanding DCMT operations originating on this node.

If no dcmt-id is specified, the vary operation will be assigned an internally generated identifier.

The identifier can subsequently be used to monitor or terminate the vary operation using DCMT DISPLAY ID and DCMT VARY ID commands.

`IMMediate`

Specifies that CA-IDMS will cancel any tasks or user sessions that prevent the vary from completing.

`PERmanent`

Specifies that the new area status is assigned permanently. This means that the status will remain in effect until it is subsequently changed by another DCMT VARY command or until the journal files are initialized.

## Usage

*Enabling data sharing for an area:* When initiating data sharing for an area, its definition must not conflict with that of any other area that is currently being accessed by a group member in the shared mode. Furthermore, if another member of the group is accessing the area in a shared mode at the time the DCMT command is issued, the definitions of the area in both systems must be identical. If these conditions are not met, a warning is issued when the DCMT VARY AREA command is issued. The status of the area cannot be changed to RETRIEVAL or UPDATE until the condition is corrected. For more information on these restrictions, refer to "Shared Area Requirements" in Chapter 3.

Additionally, before the area can be varied online, files associated with the target area must be assigned to a shared cache or there must be a default shared cache associated with the system.

None of the above restrictions apply if the area will be accessed in a transient retrieval mode.

*Changing the shared cache for a file*: In order to change or remove the shared cache assignment for a file, the following conditions must be met:

- No areas associated with the file can be in UPDATE mode
- All shared areas associated with the file must have a status of either OFFLINE or TRANSIENT RETRIEVAL

Changing the shared cache for a file affects only the system on which the command is issued. To change the shared cache for all systems that are accessing the file, the command must be issued on each of those systems. In a data sharing environment, the command can be broadcast to all members of the group. See "Broadcasting Commands" in Chapter 3.

If any area associated with the file is shared, the new shared cache will take effect only if all shared areas associated with the file have a status of OFFLINE or TRANSIENT RETRIEVAL in all group members. This is because the cache name for a file associated with a shared area (other than one in transient retrieval), is determined by the first sharing system to open the file. All systems that subsequently open the file will use the shared cache specified by the first system.

*Changing the area status permanently:* Assigning a permanent area status means that the status will remain in effect until it is changed by another DCMT VARY command or until the system's journal files are initialized. The area status will be retained across normal shutdowns and across abnormal terminations provided the area's WARMSTART option in the DMCL specifies MAINTAIN CURRENT STATUS.

The permanence of an area status has no effect on physical area locks; it only affects the mode in which the area will be accessed when the system is next started. If the CA-IDMS system is shutdown normally, all physical area locks held by the system are removed regardless of whether or not the system's area status was assigned as UPDATE PERMANENT.

*Identifying vary operations:* When changing the status of an area to RETRIEVAL, TRANSIENT RETRIEVAL, or OFFLINE, the vary operation is distinguished from other DCMT operations by means of an identifier. If one is not specified on the command, CA-IDMS will generate one as a sequential number. If the vary operation does not complete immediately, its status can be monitored using the DCMT DISPLAY ID command. The operation can be cancelled by means of the DCMT VARY ID command.

*Forcing an immediate vary:* When varying an area's status to RETRIEVAL, TRANSIENT RETRIEVAL, or OFFLINE, the change in status can be forced by specifying the IMMEDIATE option. If specified, CA-IDMS will:

- Cancel all tasks that conflict with the vary
- Terminate all user sessions with no active task if they conflict with the vary (by performing the equivalent of a DCMT VARY LTERM … RESOURCES DELETE)

- Vary offline all predefined system run units that conflict with the vary (by performing the equivalent of a DCMT VARY RUNUNIT … OFFLINE)

After the status change has occurred, predefined run units that were varied offline will be varied online unless:

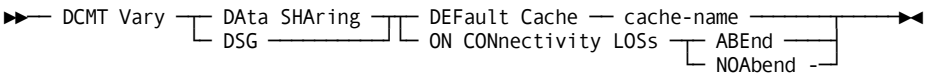
- A VARY AREA … OFFLINE was issued, since the run units will be unable to access the area
- The QUEUE run unit was varied offline, since it requires the DDLDCRUN area in update mode

If either of these conditions apply, the appropriate system run units must be varied online by explicitly issuing a DCMT VARY RUN UNIT command once their corresponding areas are again made available.

# DCMT VARY DATA SHARING

This new command provides the ability to change the default shared cache for a CA-IDMS system that is a member of a data sharing group.

## Syntax




## Parameters

`DEFault Cache` *`cache-name`*

Specifies the name of the default shared cache to be associated with the CA-IDMS system. C*ache-name* must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.

`ON CONnectivity LOSs`

Specifies what action the CA-IDMS system is to take when either a loss in connectivity to or a failure in the list or lock structure is detected. Valid values are:

- ABEnd — specifies that the CA-IDMS system is to abnormally terminate immediately.
- NOAbend — specifies that the CA-IDMS is to remain active as long as possible in order to service non-datasharing related requests.

## Usage

*Specifying a default cache*: Changes to the default shared cache will remain in effect until the system terminates or until another DCMT VARY DATA SHARING command is issued. When a system is restarted after dynamically changing the default shared cache, the default shared cache in effect will be that specified in the DMCL used by the system.

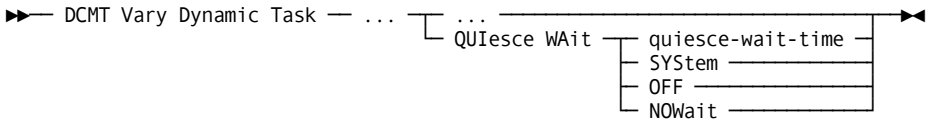
The default shared cache affects only files that have not been explicitly assigned a shared cache and for which at least one associated area is shared. The default shared cache has no affect on files that are not associated with a shared area.

*Specifying a connectivity loss option:* Changes to the connectivity loss setting will remain in effect until the system terminates or until another DCMT VARY DATA SHARING command is issued. When a system is restarted the connectivity loss specified in the DMCL used by the system will be in effect.

# DCMT VARY DYNAMIC TASK

New options on this command establish the external and quiesce wait times for a dynamically defined task.

## Syntax




## Parameters

`QUIesce WAIt`

Establishes the quiesce wait time for a task. The quiesce wait interval determines the amount of time that the task will wait on a quiesce operation before being cancelled. Valid values are:

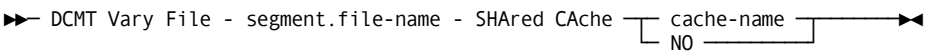
- *quiesce-wait-time*—Specifies the quiesce wait time in wall clock seconds. The value must be in the range 0 through 32,767. A value of 0 is equivalent to specifying SYSTEM.
- SYStem – Specifies that the quiesce wait time for the task is determined by the quiesce wait setting in effect for the system.
- OFF – Specifies that the task is not to be terminated due to a quiesce wait.

- NOWait — Specifies that the task is not to wait for a quiesce operation to terminate. Instead an error will be returned to the application program indicating that an area is unavailable. For navigational DML applications, this will result in an error status of 'xx66'.

# DCMT VARY FILE

This command has been revised to remove AVAILABLE as an option on the SHARED CACHE parameter. Refer to the topic, "Changing a System's Default Cache," in Chapter 3, “Exploiting the Parallel Sysplex Architecture.”

## Syntax




## Parameters

`SHAred CAche`

Specifies the name or status of shared cache for the named file. Valid values are:

- *cache-name*—Specifies that the file is to be assigned to the named cache structure. Cache-name must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.
- NO—Specifies that the file is no longer assigned to a cache structure.

## Usage

*Changing the shared cache for a file*: In order to change or remove the shared cache assignment for a file, the following conditions must be met:

- No areas associated with the file can be in UPDATE mode
- All shared areas associated with the file must have a status of either OFFLINE or TRANSIENT RETRIEVAL

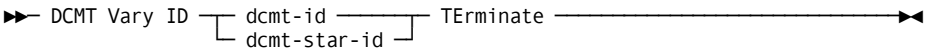
Changing the shared cache for a file affects only the system on which the command is issued. To change the shared cache for all systems that are accessing the file, the command must be issued on each of those systems. In a data sharing environment, the command can be broadcast to all members of the group. See the topic, "Broadcasting Commands," in Chapter 3, "Exploiting the Parallel Sysplex Architecture."

If any area associated with the file is shared, the new shared cache will take effect only if all shared areas associated with the file have a status of OFFLINE or TRANSIENT RETRIEVAL in all group members. This is because the cache name for a file associated with a shared area (other than one in transient retrieval), is determined by the first sharing system to open the file. All systems that subsequently open the file will use the shared cache specified by the first system.

# DCMT VARY ID

This new command terminates an outstanding DCMT request.

## Syntax




## Parameters

`ID`

Identifies the DCMT operations to be terminated.

- *dcmt-id* – Specifies the identifier of the DCMT operation to be terminated.
- *dcmt-star-id* – Specifies that all DCMT operations whose identifier begins with the specified alphanumeric characters be terminated. Dcmt-star-id is a character string whose last character is an asterisk (*) that denotes a wild card character.

  In this example, CA-IDMS will terminate all DCMT operations whose identifier begins with CUST:

dcmt v id cust* terminate

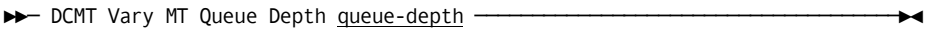
## Usage

*Referencing DCMT operations in a Data Sharing Environment:* In a data sharing environment, the DCMT VARY ID command must execute on the same member as that on which the target operation originated.

# DCMT VARY MT

This new command changes the multitasking queue depth for the system.

## Syntax



### Parameters

*queue-depth*

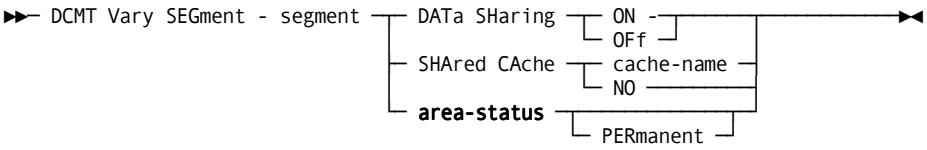
Specifies the depth of the multitasking queue.

# DCMT VARY SEGMENT

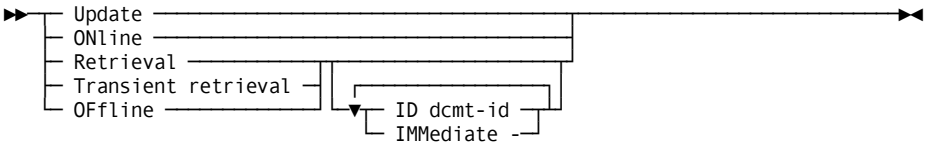
This command has been revised in the following ways:

- The sharability state of a segment can now be changed
- The semantics associated with the shared cache parameter have been revised slightly and "AVAILABLE" has been removed as an option
- A change in the status of the segment's areas can now be retained across system shutdowns
- An option has been added to specify that conflicting tasks should be cancelled if they prevent an area associated with the segment from being varied
- Certain vary segment commands can be cancelled by means of the DCMT VARY ID command

### Syntax




Expansion of **area-status**



## Parameters

`DATa SHaring`

Specifies the sharability state of all areas in the named segment. The change will be made only to areas whose status is OFFLINE. Valid values are:

- ON—Specifies that this system is eligible to share update access to all areas of the named segment with other members of the system's data sharing group.
- OFf—Specifies that this system will no longer be eligible to share update access to the areas of the named segment with other members of the system's data sharing group.

`SHAred CAche`

Specifies the name or status of shared cache for all files in the named segment. Valid values are:

- *cache-name*—Specifies that all files associated with the named segment are to be assigned to the named cache structure. C*ache-name* must identify an XES cache structure defined to a coupling facility accessible to the CA-IDMS system.
- NO—Specifies that the files associated with the named segment are no longer assigned to a cache structure.

**`area-status`**

Specifies a new status to be assigned to the areas associated with the named segment. For valid options refer to the *CA-IDMS System Tasks and Operator Commands*.

`dcmt-id`

Specifies the identifier that is to be assigned to this vary operation. *Dcmt-id* must be a 1 – 8 alphanumeric character string that is unique across all outstanding DCMT operations originating on this node.

If no dcmt-id is specified, the vary operation will be assigned an internally generated identifier.

The identifier can subsequently be used to monitor or terminate the vary operation using DCMT DISPLAY ID and DCMT VARY ID commands.

`IMMediate`

Specifies that CA-IDMS will cancel any tasks or user sessions that prevent the vary from completing.

PERmanent

Specifies that the new area status is to be assigned permanently. This means that the status will remain in effect until it is subsequently changed with another DCMT VARY command or until the journal files are initialized.

## Usage

*Enabling data sharing for an area:* When initiating data sharing for an area, its definition must not conflict with that of any other area that is currently being accessed by a group member in the shared mode. Furthermore, if another member of the group is accessing the area in a shared mode at the time the DCMT command is issued, the definitions of the area in both systems must be identical. If these conditions are not met, a warning is issued when the DCMT VARY AREA command is issued. The status of the area cannot be changed to RETRIEVAL or UPDATE until the condition is corrected. For more information on these restrictions, refer to "Shared Area Requirements" in Chapter 3.

Additionally, before the area can be varied online, files associated with the target area must be assigned to a shared cache or there must be a default shared cache associated with the system.

None of the above restrictions apply if the area will be accessed in a transient retrieval mode.

*Changing the shared cache for a file*: In order to change or remove the shared cache assignment for a file, the following conditions must be met:

- No areas associated with the file can be in UPDATE mode
- All shared areas associated with the file must have a status of either OFFLINE or TRANSIENT RETRIEVAL

Changing the shared cache for a file affects only the system on which the command is issued. To change the shared cache for all systems that are accessing the file, the command must be issued on each of those systems. In a data sharing environment, the command can be broadcast to all members of the group. See "Broadcasting Commands" in Chapter 3.

If any area associated with the file is shared, the new shared cache will take effect only if all shared areas associated with the file have a status of OFFLINE or TRANSIENT RETRIEVAL in all group members. This is because the cache name for a file associated with a shared area (other than one in transient retrieval), is determined by the first sharing system to open the file. All systems that subsequently open the file will use the shared cache specified by the first system.

*Changing the area status permanently:* Assigning a permanent area status means that the status will remain in effect until it is changed by another DCMT VARY command or until the system's journal files are initialized. The area status will be retained across normal shutdowns and across abnormal terminations provided the area's WARMSTART option in the DMCL specifies MAINTAIN CURRENT STATUS.

The permanence of an area status has no effect on physical area locks; it only affects the mode in which the area will be accessed when the system is next started. If the CA-IDMS system is shutdown normally, all physical area locks held by the system are removed regardless of whether or not the system's area status was assigned as UPDATE PERMANENT.

*Identifying vary operations:* A vary segment command is translated into a set of vary area operations, one for each area associated with the segment. When changing the status of a segment to RETRIEVAL, TRANSIENT RETRIEVAL, or OFFLINE, each vary area operation is assigned an identifier. If a dcmt-id is specified on the vary segment command, it is used to generate the identifiers for the associated vary area operations. If no dcmt-id is specified, each vary area operation is identified by a unique number.

In order to generate the identifier if a dcmt-id is specified, CA-IDMS appends a sequential number to the dcmt-id, truncating the dcmt-id value if necessary. The following examples illustrate the identifiers that are generated for different dcmt-id values.

- A dcmt-id of CUST results in identifiers of CUST0001, CUST0002, etc.
- A dcmt-id of CUSTOMER results in identifiers of CUSTOME1, CUSTOME2, etc.

The vary segment operation returns an error if the generated identifier of any vary area operation would be the same as the identifier of another outstanding DCMT operation.

*Forcing an immediate vary:* When varying a segment's status to RETRIEVAL, TRANSIENT RETRIEVAL, or OFFLINE, the change in status can be forced by specifying the IMMEDIATE option. If specified, CA-IDMS will:

- Cancel all tasks that conflict with the vary
- Terminate all user sessions with no active task if they conflict with the vary (by performing the equivalent of a DCMT VARY LTERM … RESOURCES DELETE)
- Vary offline all predefined system run units that conflict with the vary (by performing the equivalent of a DCMT VARY RUNUNIT … OFFLINE)

After the status change has occurred, predefined run units that were varied offline will be varied online unless:

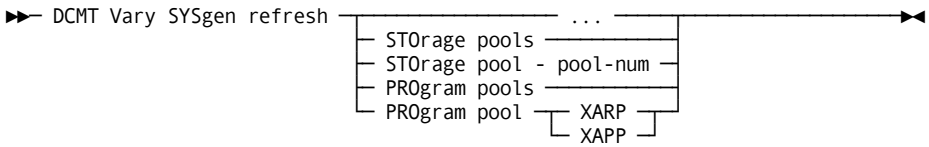
- A VARY AREA … OFFLINE was issued, since the run units will be unable to access the area
- A VARY SEGMENT … OFFLINE was issued, since it requires the DDLDCRUN area in update mode

If either of these conditions apply, the appropriate system run units must be varied online by explicitly issuing a DCMT VARY RUN UNIT command once their corresponding areas are again made available.

# DCMT VARY SYSGEN REFRESH

This command has been revised to apply pending sysgen changes for storage and program pools.

## Syntax




## Parameters

`STOrage pools`

Specifies that sysgen changes for all XA storage pools should be applied.

`STOrage pool` *`pool-num`*

Specifies that sysgen changes for the specified XA storage pool should be applied.

*pool-num*

Identifies the number of the storage pool for which sysgen changes should be applied.

`PROgram pools`

Specifies that sysgen changes for all XA program pools should be applied.

`PROgram pool XARP/XAPP`

Specifies that sysgen changes for the specified program pool should be applied.

- XARP—Indicates that sysgen changes for the XA reentrant program pool should be applied.
- XAPP—Indicates that sysgen changes for the XA non-reentrant program pool should be applied.

## Usage

*Dynamic sysgen changes for program pools:* The following types of program pool changes can be applied dynamically:

- The addition of a new XA program pool
- The increase in size of an existing XA program pool

The successful application of a dynamic change to a program pool depends on the availability of space to effect the change.

*Dynamic sysgen changes for storage pools:* The following types of storage pool changes can be applied dynamically:

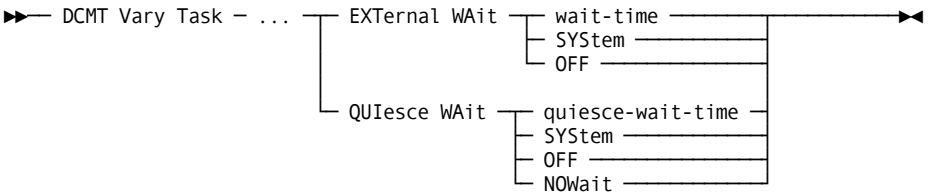
- The addition of a new XA storage pool
- The increase in size of an existing XA storage pool

The successful application of a dynamic change to a storage pool depends on the availability of space to effect the change.

# DCMT VARY TASK

New options on this command permit dynamically changing the external or quiesce wait times for a task.

## Syntax



## Parameters

`EXTernal WAIt`

Varies the external wait setting for a task. Valid values include:

- *wait-time*—The new external wait time in seconds. The value must be in the range 0 through 32,767. A value of 0 is equivalent to specifying SYSTEM.
- SYStem—Indicates that the external wait time for a task is to be set to the value established for the system.
- OFF—Indicates that there is no limit to the length of time that the system will wait for an external user session to issue a database request.

`QUIesce WAIt`

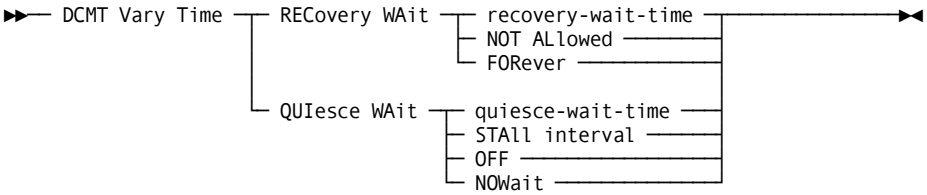
Varies the quiesce wait time for a task. Valid values include:

- *quiesce-wait-time*— Specifies the new quiesce wait time in wall clock seconds. The value must be in the range 0 through 32,767. A value of 0 is equivalent to specifying SYSTEM.
- SYStem – Specifies that the quiesce wait time for the task is determined by the quiesce wait setting in effect for the system.
- OFF – Specifies that the task is not to be terminated due to a quiesce wait.
- NOWait — Specifies that the task is not to wait for a quiesce operation to terminate. Instead an error will be returned to the application program indicating that an area is unavailable. For navigational DML applications, this will result in an error status of 'xx66'.

# DCMT VARY TIME

New options on this command permit dynamically changing the recovery or quiesce wait setting for the system.

## Syntax



## Parameters

`RECovery WAit`

Varies the recovery wait setting. The recovery wait setting is initially established at system generation time by the RECOVERY WAIT parameter of the SYSTEM statement. Valid values include:

- *recovery-wait-time*—The new recovery wait time in wall clock seconds. The value must be in the range 0 through 32,767. A value of 0 is equivalent to specifying NOT ALLOWED.
- NOT ALlowed—Indicates that tasks will not wait for the recovery of resources by failed data sharing group members; instead they will be aborted.
- FORever—Indicates that tasks will wait indefinitely for the recovery of resources by failed data sharing group members.

`QUIesce WAIt`

Varies the quiesce wait time for the system. The quiesce wait time determines the amount of time that a task will wait on a quiesce operation before being cancelled. Valid values are:

- *quiesce-wait-time*— Specifies the new quiesce wait time in wall clock seconds. The value must be in the range 0 through 32,767. A value of 0 is equivalent to specifying STALL INTERVAL
- STAll interval – Specifies that the quiesce wait time for a task is the same as its stall interval.
- OFF – Specifies that tasks are not to be terminated due to quiesce waits.
- NOWait — Specifies that tasks are not to wait for a quiesce operation to terminate. Instead an error will be returned to the application program indicating that an area is unavailable. For navigational DML applications, this will result in an error status of 'xx66'.

# DCMT Command Codes

The following command codes apply to new and revised DCMT commands available in Release 15.0.

| Code | DCMT Command |
|---|---|
| N009 | VARY AREA |
| N009019 | Remove entry for SHARED CACHE AVAILABLE |
| N009020 | DATA SHARING ON |
| N009021 | DATA SHARING OFF |
| N024 | VARY TASK |
| N024019 | EXTERNAL WAIT |
| N024020 | QUIESCE WAIT |
| N026 | VARY TIME |
| N026006 | RECOVERY WAIT |
| N026007 | QUIESCE WAIT |
| N035 | HELP |
| N035043 | DATA SHARING |
| N035044 | ID |
| N035045 | DBNAME (DBTABLE) |
| N076 | DISPLAY SUBTASK/MT |
| N076004 | MT QUEUE DEPTH |
| N077 | VARY SUBTASK/MT |
| N077003 | MT QUEUE DEPTH number |
| N082 | VARY FILE |
| N082019 | Remove entry for SHARED CACHE AVAILABLE |
| N091 | VARY SEGMENT |
| N091019 | Remove entry for SHARED CACHE AVAILABLE |
| N091020 | DATA SHARING ON |
| N091021 | DATA SHARING OFF |

| Code | DCMT Command |
|---|---|
| N093 | DISPLAY LOCKS |
| N093005 | STATISTICS |
| N093006 | RECORD DATA |
| N093007 | RECORD DATA MEMBER member-name |
| N094 | DISPLAY SYSGEN REFRESH |
| N094004 | STORAGE POOL |
| N094005 | STORAGE POOL pool-number |
| N094006 | PROGRAM POOL |
| N094007 | PROGRAM POOL pool-name |
| N095 | VARY SYSGEN REFRESH |
| N095004 | STORAGE POOL |
| N095005 | STORAGE POOL pool-number |
| N095006 | PROGRAM POOL |
| N095007 | PROGRAM POOL pool-name |
| N096 | DISPLAY/VARY DATA SHARING |
| N096030 | DISPLAY DATA SHARING SUMMARY |
| N096031 | DISPLAY DATA SHARING XES LOCK |
| N096032 | DISPLAY DATA SHARING XES LIST |
| N096033 | DISPLAY DATA SHARING XES GROUP |
| N096034 | DISPLAY DATA SHARING ALL |
| N096036 | VARY DATA SHARING DEFAULT CACHE cache-name |
| N098 | QUIESCE |
| N098001 | AREA |
| N098002 | SEGMENT |
| N098003 | DBNAME |
| N099 | DISPLAY ID |
| N099001 | DISPLAY ID |
| N099002 | DISPLAY ID dcmt-id |
| N101 | VARY ID |
| N101003 | VARY ID dcmt-id |

Appendix

B

# New and Revised User Exits

This chapter describes new and revised user exits available in Release 15.0.

## OPTIXIT

This exit is invoked whenever an IDMS request unit is initiated within the CICS environment. It allows the OPTI structure to be dynamically modified so that an individual request unit is routed to a specific backend CV.

### Considerations

This exit is passed information about an IDMS request and where it is to be routed by default. The exit may update the routing information in order to direct it to another location.

The first and third parameters passed to the exit contain information about the request: the address of the subschema control block and the caller's registers at the time the request was issued, respectively.

The second and fourth parameters contain routing information. The second parameter is the address of an OPTI structure generated either from parameters in the CICSOPT macro or from the SYSCTL file specified in the CICSOPT SYSCTL parameter. The fourth parameter may contain addresses of OPTI structures that were generated from additional SYSCTL files, if the MAXCVNO parameter was specified in the CICSOPT macro. The fourth parameter is always passed, even if no MAXCVNO was specified.

In order to direct the request unit to a different backend CV, the exit must update the OPTI structure passed in the second parameter. It may use information passed in the fourth parameter to determine which backend CVs are available.

## Registers on Entry

When the exit is given control, the contents of the registers are:

- **Register 1** – points to a four-word parameter list described below.
- **Register 13** – points to a savearea
- **Register 14** – points to the return address
- **Register 15** – points to the entry point of the exit

## Parameters

Four parameters are passed to the exit:

- **Fullword 1** – The address of the application program's subschema control block (SSC)
- **Fullword 2** – The address of an OPTI structure describing where the request will be routed unless overridden by the exit. (The OPTI structure is described by dsect #OPIDS.)
- **Fullword 3** – The address of a savearea where the registers at entry to the interface are saved
- **Fullword 4** – the address of an array of fullwords whose contents are:
  - Fullword 1 - The MAXCVNO value specified in the CICSOPT macro. This value indicates the number of additional OPTI addresses present in this array.
  - Fullwords 2-4 - Three words that are available for use by the exit. The contents of these fields are preserved across invocations of the OPTIXIT (and the OPTIQXIT if present). These three words can be used to retain information such as the last CV to which a request unit was routed. On the first call to the exit, the value of the first word will be the address of the default OPTI structure (see Fullword 5); the value of the remaining two words will be zero.
  - Fullword 5 - The address of the default OPTI structure. This is the OPTI structure generated from the CICSOPT macro or from the file whose DDNAME is specified by the SYSCTL parameter of the CICSOPT macro.
  - Remaining Fullwords - Addresses of the OPTI structures generated from additional SYSCTL files as determined by the MAXCVNO value. If the MAXCVNO value is 0, no additional addresses are present in the array.

# OPTIQXIT

This exit is invoked whenever an IDMS SQL session is initiated within the CICS environment. It allows the default OPTI structure to be dynamically modified so that an individual SQL session is routed to a specific backend CV.

## Considerations

This exit is passed information about the current IDMS SQL request and where it is to be routed by default. The exit may update the routing information in order to direct the SQL session to another location.

The first, third, fifth, and sixth parameters contain information about the SQL request and the application program issuing the request.

The second and fourth parameters contain routing information. The second parameter is the address of an OPTI structure generated either from parameters in the CICSOPT macro or from the SYSCTL file specified in the CICSOPT SYSCTL parameter. The fourth parameter may contain addresses of OPTI structures that were generated from additional SYSCTL files if the MAXCVNO parameter was specified in the CICSOPT macro. The fourth parameter is always passed, even if no MAXCVNO was specified.

In order to direct the SQL session to a different backend CV, the exit must update the OPTI structure passed in the second parameter. It may use information passed in the fourth parameter to determine which backend CVs are available.

## Registers on Entry

When the exit is given control, the contents of the registers are:

- **Register 1** – points to a six-word parameter list described below
- **Register 13** – points to a savearea
- **Register 14** – points to the return address
- **Register 15** – points to the entry point of the exit

## Parameters

Six parameters are passed to the exit:

- **Fullword 1** – The address of the application program's SQL information block (#SQLPIB dsect)
- **Fullword 2** – The address of an OPTI structure describing where the session will be routed unless overridden by the exit. (The OPTI structure is described by dsect #OPIDS.)
- **Fullword 3** – The address of a savearea where the registers at entry to the interface are saved
- **Fullword 4** – the address of an array of fullwords whose contents are:
  - Fullword 1 - The MAXCVNO value specified in the CICSOPT macro. This value indicates the number of additional OPTI addresses present in this array.

  - Fullwords 2-4 - Three words that are available for use by the exit. The contents of these fields are preserved across invocations of the OPTIQXIT (and the OPTIXIT if present). These three words can be used to retain information such as the last CV to which a session was routed. On the first call to the exit, the value of the first word will be the address of the default OPTI structure (see Fullword 5); the value of the remaining two words will be zero.
  - Fullword 5 - The address of the default OPTI structure. This is the OPTI structure generated from the CICSOPT macro or from the file whose DDNAME is specified by the SYSCTL parameter of the CICSOPT macro.
  - Remaining Fullwords - Addresses of the OPTI structures generated from additional SYSCTL files as determined by the MAXCVNO value. If the MAXCVNO value is 0, no additional addresses are present in the array.

- **Fullword 5** - The SQL command information block (dsect #SQLCIB) that describes the statement being executed.
- **Fullword 6** - The contents of this field depend on the type of SQL statement being executed as indicated by the SQCIBCMD field of the command information block (Fullword 5).
  - If the SQL statement being executed by the application is a CONNECT (SQCIBCMD value 7), then Fullword 6 contains the address of the dictionary name to which the SQL session is connecting.
  - If the SQL statement being executed by the application is an EXECUTE IMMEDIATE (SQCIBCMD value 14), then Fullword 6 contains the address of the text string representing the statement to be executed. If the text represents a CONNECT statement, the dictionary name to which the session is connecting is also in the text string. (The length of the text string is in SQCIBCML.)
  - If the SQL statement being executed is neither a CONNECT nor an EXECUTE IMMEDIATE, Fullword 6 is zero.

# Exit 35 – Stalled Task Information Exit

This exit is called to gather information about a stalled task for use during deadlock victim selection. It will be invoked only in a data sharing environment during global deadlock management.

This exit can be used to pass information to user exit 36 to assist in selecting a victim task in a global deadlock situation.

## Considerations

This exit is passed the addresses of two control blocks. The first control block contains information about the stalled task, the second is an output area in which the exit can place information for use by exit 36 when selecting a deadlock victim. The address of the stalled task's Dispatch Control Element (DCE) is passed in the first control block. This can be used to locate other task-related control blocks.

Addresses should not be stored in the 32-byte output area, since exit 36 may execute on a different system from that on which exit 35 is executing.

This exit must be written to execute in SYSTEM mode. The #DEFXIT macro that adds the exit routine to the system must specify:

- MODE=SYSTEM
- AMODE=ANY

Additionally, this exit should be reentrant and should be coded to handle 31-bit addresses.

## Parameters

Two parameters are passed:

- **Fullword 1** – The address of an area described by dsect #X35PL
- **Fullword 2** – The address of a 32-byte output area in which the exit may save information to be passed to User Exit 36

## Return Codes

Set register 15 to 0.

# Exit 36 – Global Deadlock Victim Selection Exit

This exit is called in a global deadlock situation to select a task to be cancelled. If exit 36 is not installed, a global deadlock will be resolved by choosing the task with the lowest priority that was initiated last. When comparing two tasks, CA-IDMS will always select as a victim a task that specified COND=DEAD over one that specified COND=NONE.

Exit 36 allows site-control over which task is chosen as a victim in a global deadlock situation.

## Considerations

This exit is passed the addresses of two sets of control blocks, each of which is associated with a stalled task. Within each set, the first control block contains information supplied by CA-IDMS about the stalled task, the second is an area that potentially contains information passed from exit 35.

Exit 36 must be written to execute in SYSTEM mode. The #DEFXIT macro that adds the exit routine to the system must specify:

- MODE=SYSTEM
- AMODE=ANY

Additionally, this exit should be reentrant and should be coded to handle 31-bit addresses.

## Parameters

Four parameters are passed:

- **Fullword 1** – The address of a control block described by dsect #X36PL that describes the first of two deadlocked tasks.
- **Fullword 2** – The address of a 32-byte area containing information passed from user exit 35 for the first deadlocked task.
- **Fullword 3** – The address of a control block described by dsect #X36PL that describes the second of two deadlocked tasks.
- **Fullword 4** – The address of a 32-byte area containing information passed from user exit 35 for the second deadlocked task.

## Return Codes

Set register 15 to the address of the control block described by dsect #X36PL for the task that is to be selected as the victim. If the exit does not select a victim, it should set the return code to 0.

# Exit 37 – Recovery Wait Exit

This exit is called when a task is about to wait on a global resource that requires recovery by a failed member of a data sharing group.

This exit can override the current recovery wait setting for the system.

### Considerations

The exit can specify whether the task should be aborted or whether it should wait for the failing member to be recovered. If the task should wait, the exit specifies the length of time the task should wait.

This exit must be written to execute in SYSTEM mode. The #DEFXIT macro that adds the exit routine to the system must specify:

- MODE=SYSTEM
- AMODE=ANY

Additionally, this exit should be reentrant and should be coded to handle 31-bit addresses.

### Parameters

One parameter is passed:

Fullword 1 – The address of an area described by dsect #X37PL

### Return Codes

By setting an appropriate return code in Register 15, the exit can specify what action CA-IDMS should take with regard to the task . The possible choices are:

- 0 – the task should wait. Register 0 must contain the amount of time that the task is permitted to wait. Valid values for register 0 are:
  - 1 through 32767 specifying the number of seconds that the task is permitted to wait
  - -1 indicating that the task should wait indefinitely
  - 0 indicating that the task should not wait. A value of 0 is equivalent to a return code value of 8.
- 4 – the system's recovery wait setting determines what action will be taken
- 8 – the task should be aborted

## Exit 38 – Quiesce Area Exit

This exit is invoked when a quiesce point has been reached in the processing of a DCMT QUIESCE command. Its purpose is to allow additional site-specific actions to be taken in response to the quiesce.

## Considerations

This exit is passed the quiesce identifier, an indication of what is being quiesced and a list of files and their dataset names that are impacted by the quiesce. With this information, the exit can take additional action, such as constructing JCL or loading predefined JCL for a batch job to be submitted through the internal reader.

Through return codes, the exit can direct IDMS to terminate or continue the quiesce operation, or proceed as specified in the original DCMT QUIESCE command.

## Parameters

The exit is passed a single parameter described by DSECT #X38PL. This structure contains the following information:

- The nodename on which the quiesce command originated
- The quiesce operation identifier
- An indication of what is being quiesced (area, segment or DBNAME) and its name
- An array of file entries containing the following information for each file involved in the quiesce:
  - File name (<segment-name>.<area-name>)
  - VOLSER
  - DDNAME
  - Data set name

## Return Codes

By setting an appropriate return code, the exit can specify what action IDMS should take with regard to the quiesce operation. The possible choices are:

- 0, to continue or terminate the quiesce operation as specified on the DCMT QUIESCE command
- 8, to continue the quiesce operation, overriding the option specified on the DCMT QUIESCE command
- 12, to terminate the quiesce operation, overriding the option specified on the DCMT QUIESCE command

Appendix

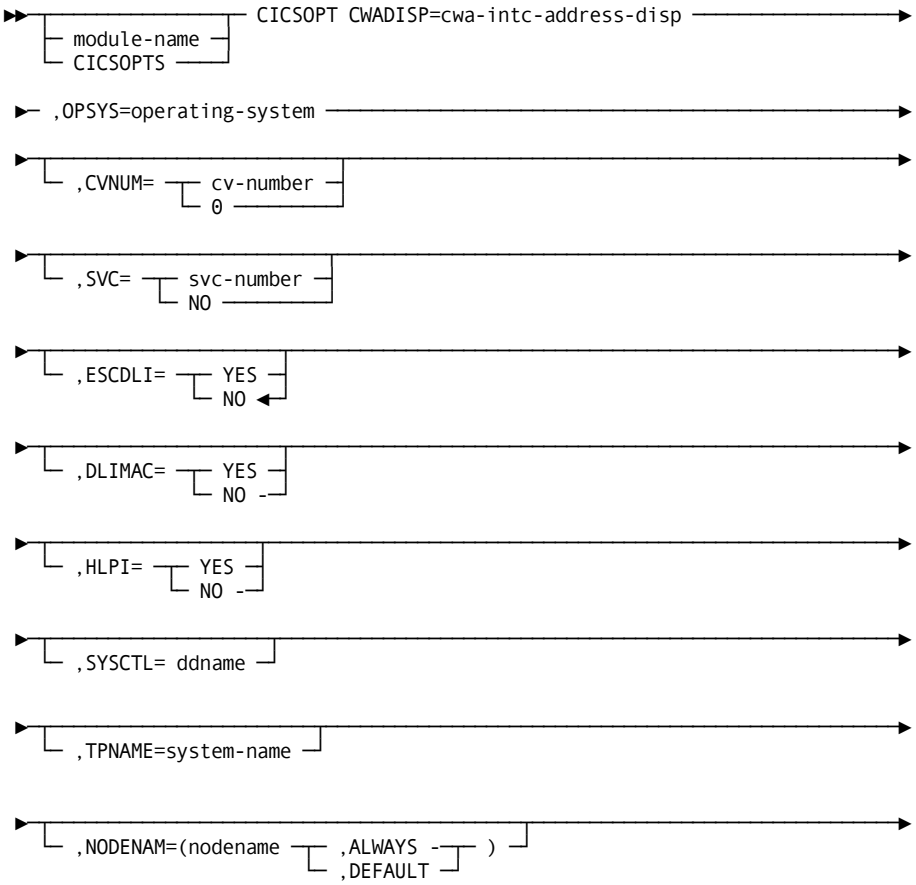
C

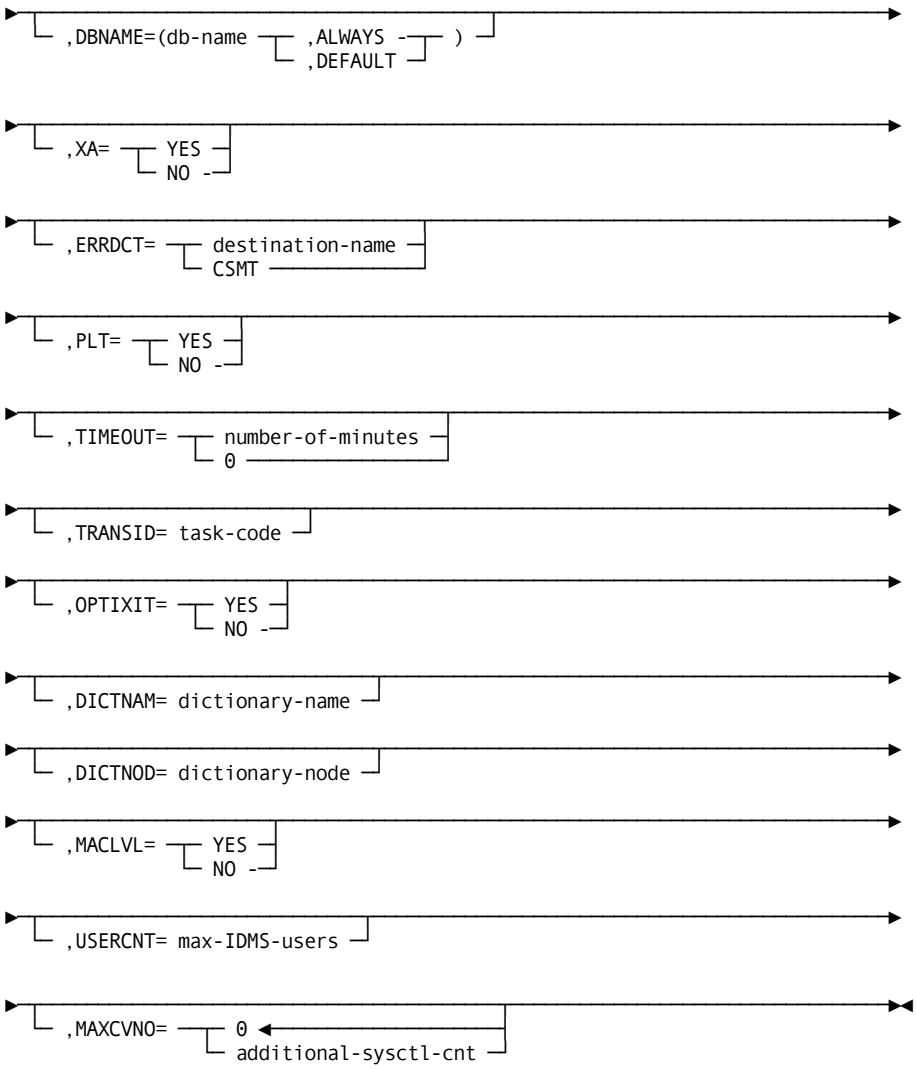
# CICSOPT Macro

## CICSOPT Macro

The CICSOPT macro is used to tailor the new CICS interface. This appendix describes the parameters that may be specified for the CICSOPT macro and instructions on its assembly and the creation of the IDMSINTC interface module.

### Syntax






## Parameters

`module-name`

Identifies the CSECT name of the generated module. Default is CICSOPT.

`CWADISP`

Identifies the displacement within the CICS CWA of a fullword to hold the address of the CICSOPT module.

**cwa-intc-address-disp** must specify either an integer in the range 0 through 3584 representing an offset within the CWA or the name of a field within the CSA copy book.

The value specified must begin on a fullword boundary and be the same value that is specified in the CWADISP parameter of the IDMSCINT macro.

`OPSYS`

Identifies the operating system under which the DC/UCF (or CICS) system will run. Valid values include:

- MVS
- VSE
- DVS
- DS
- DOS
- DOSVS

`CVNUM`

Identifies the number of the DC/UCF system to be accessed from CICS.

For **cv-number**, specify the number used for the CVNUM parameter in the CA-IDMS system definition.

`SVC`

Identifies the number of the CA-IDMS SVC. For **svc-number**, specify a value as follows:

- If no SVC is being used, or if using SYSCTL, specify NO.
- If an SVC is being used by the DC/UCF system, specify the SVC number.

The SVC parameter is required if no SYSCTL file is specified.

`ESCDLI`

Is never explicitly specified. CAISAG for OS/390, or CAIIJMP for VSE/ESA and BS200/OSD automatically sets this parameter to YES if installing CA-IDMS/DLI Transparency and the CICS interface (INT-CICS).

`DL1MAC`

Always NO. Provided solely for upward compatibility.

`HLPI`

Specifies whether or not HLPI support is required for DL1.

SYSCTL

Identifies the ddname of the file containing DC/UCF system control information.

If no SVC (described above) is specified, the SYSCTL parameter is required. Likewise, if SYSCTL is desired, the SVC parameter must be NO (SVC=NO).

TPNAME

Specifies the name by which DC/UCF will identify all tasks running under this CICS system. For system-name, specify a four-character name.

If TPNAME is omitted or system-name is specified as spaces, system-name defaults to the four-character local CICS system id.

This name forms the first part of the local transaction ID for database requests. It also forms the first four characters of the front-end system ID for external request units. "BULK" is appended to system-name to form the front-end system ID. The front-end system ID is used in determining the packet size for communications and may also be used as an alternate task code for controlling external request unit processing.

NODENAM

Identifies a system defined to the DC/UCF communications network to be contained in the CICSOPT module and the conditions under which programs signing on to the DC/UCF system will be directed to the named node for execution. This parameter has no effect on SQL database sessions. For SQL sessions, refer to the DICTNOD parameter.

For nodename, specify the one- to eight-character name of a remote system. If the node name is not specified, the DC/UCF obtains the appropriate node name from the application program or from the SYSCTLfile (OS/390 only).

ALWAYS

Indicates that nodename is to override any node named by the program. Requests from programs signing on to DC/UCF are always directed to the named node regardless of node name specifications made by the program.

DEFAULT

Indicates that requests from programs signing on to DC/UCF are to be directed to the named node only if the program does not name a node.

**Note:** Under OS/390 and VSE/ESA, SYSCTL node name specifications can override CICSOPT and program specifications.

DBNAME

Identifies the database (or data dictionary) name to be contained in the CICSOPT module. This parameter also identifies the conditions under which programs signing on to the DC/UCF system access the named database. This parameter has no effect on SQL database sessions. For SQL sessions, refer to the DICTNAM parameter.

For db-name, specify the name of the database that programs are to access when running under the DC/UCF system. If the database name is not specified, DC/UCF obtains the appropriate database name from the application program or from the SYSCTL file (OS/390 only).

ALWAYS

Indicates that db-name is to override any database named by the program. Programs signing on to DC/UCF always execute against the named database regardless of database name specifications made by the program.

DEFAULT

Indicates that programs signing on to DC/UCF are to execute against the named database only if the program does not name a database.

**Note:** Under OS/390 and VSE/ESA, SYSCTL database name specifications can override CICSOPT and program specifications.

XA=NO/YES

Designates whether the operating system is XA (YES) or not (NO). If you specify YES, IDMSINTC allocates the primary user-oriented storage in the 31-bit storage area. This storage is retained across all successful task terminations for terminal-associated tasks, and this storage is reused on the next DC/UCF request. The storage is freed for any failing or non-terminal task.

MACLVL=YES/NO

Indicates whether applications using the CICS macro-level interface will be supported by this CA-IDMS interface.

**Caution:** If a macro-level application attempts communications with a CA-IDMS interface assembled with MACLVL=NO, the results will be unpredictable.

ERRDCT

Identifies the CICS transient data destination to be used as the target for error messages produced by IDMSINTC and IDMSTRUE. The default destination-name is CSMT. Use another destination if you want to route CA-IDMS error messages to another CICS destination. The DCT entry should be defined with a logical record length of at least 130 characters.

`PLT=YES/NO`

Indicates how IDMSINTC starts up. YES indicates that IDMSINTC can start up as a PLT-invoked program. NO indicates IDMSINTC always starts up as a user task once CICS start up is complete.

`TIMEOUT`

Indicates the number of minutes (0 to 10) before which automatic removal of storage occurs. If you specify a nonzero value, IDMSINTC generates a return code of 4 with a warning message that any application code using the TS queue SET option will not function properly. Specify a nonzero value to conserve CICS storage. The default is 0.

`TRANSID`

Identifies the transaction coded in the program control table (PCT) as invoking IDMSINTC. Task-code must be the name of a task defined in the PCT table.

`OPTIXIT=YES/NO`

Indicates whether CICS transactions can modify the IDMSOPTI structure dynamically so that only the task thread is affected by the changes. YES indicates that the IDMSOPTI structure can be modified dynamically.

IDMSINTC copies the static IDMSOPTI structure into dynamic storage and passes it to the user routine, which may alter it based on site-specific rules.

`DICTNAM`

Sets the dictionary to which an SQL session will be connected unless it is overridden by the application program. This parameter has no effect on non SQL database sessions.

`DICTNOD`

Sets the node to which an SQL session will be connected. This parameter has no effect on non SQL database sessions.

`USERCNT`

Specifies the maximum number of CICS users that can be using IDMS through this interface. This includes all active sessions, all suspended sessions and all users that have not yet timed out (see TIMEOUT). Valid values are in the range 1 to 100000. The default value is 100.

MAXCVNO

Specifies the number of additional SYSCLT DD cards that can be accessed through this interface. DDNAMEs for the additional SYSCTL files are derived by replacing either the first blank or the last character (if all 8-bytes are in use) of the SYSCTL operand value by the numbers 1 through MAXCVNO.

## Assembling CICSOPT and Link Editing IDMSINTC

JCL to assemble CICSOPT and link edit the IDMSINTC module appears below for OS/390 and VSE/ESA operating systems.

**Note:** The following JCL does not use SMP/E. For examples of how to apply a modification to a CA-IDMS load library using SMP/E, see the SAMPJCL library delivered with the CA-IDMS installation tape.

### CICSOPTS Assembly and IDMSINTC Link Edit (OS/390)

```
//*-------------------------------------------------------------------
//*            ASSEMBLER IEV90 JOB STREAM
//*-------------------------------------------------------------------
//ASMSTEP EXEC PGM=IEV90,
//        PARM='ALIGN,XREF,PUNCH,NODECK',
//        REGION=2048K
//SYSLIB       DD  DSN=idms.maclib,DISP=SHR
//             DD  DSN=idms.srclib,DISP=SHR
//             DD  DSN=cics.maclib,DISP=SHR
//             DD  DSN=mvs.maclib,DISP=SHR
//SYSUT1       DD  DSN=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT2       DD  DSN=&&SYSUT2,UNIT=VIO,SPACE=(1700,(600,100))
//SYSUT3       DD  DSN=&&SYSUT3,UNIT=VIO,SPACE=(1700,(600,100))
//SYSPRINT     DD  SYSOUT=*
//SYSPUNCH     DD  DSN=&&OBJECT,
                     DISP=(NEW,PASS),
                     UNIT=SYSDA,
                     SPACE=(80,(500,1000))
//SYSIN        DD  *
CICSOPT macro input parameters
  END
//*-------------------------------------------------------------------
//*            LINK IEWL
//*-------------------------------------------------------------------
//LINK    EXEC PGM=IEWL,
//        PARM='LET,LIST,XREF,RENT',
//        REGION=128K,
//        COND=(8,LT,ASMSTEP)
//SYSLMOD      DD  DSN=idms.loadlib,DISP=SHR
//SYSPRINT     DD  SYSOUT=*
//SYSUT1       DD  DSN=&&SYSUT1,
//             UNIT=SYSDA,
//             SPACE=(6400,(80)),
//             DISP=(NEW,PASS)
//IN1          DD  DSN=idms.distload,DISP=SHR
//IN2          DD  DSN=cics.loadlib,DISP=SHR
//IN3          DD  DSN=&&OBJECT,DISP=(OLD,DELETE)
//SYSLIN       DD  DDNAME=SYSIN
//SYSIN        DD  *
 ORDER DFHEAI
```

```
INCLUDE IN2(DFHEAI)
INCLUDE IN3
INCLUDE IN1(INTC0410)
INCLUDE IN1(IDMSTRUE)
INCLUDE IN1(IDMS)
INCLUDE IN2(DFHEAIO)

ENTRY STARTUP
MODE AMODE(31),RMODE(24)
NAME IDMSINTC(R)
```

| Item | Description |
|---|---|
| cics.loadlib | data set name of the CICS load library |
| cics.maclib | data set name of the CICS macro library |
| idms.distload | data set name of the CA-IDMS SMP/E distribution load library |
| idms.loadlib | data set name of the CA-IDMS load library |
| idms.maclib | data set name of the CA-IDMS macro library |
| idms.srclib | data set name of the CA-IDMS source library |
| mvs.maclib | data set name of the OS/390 system macro library |

### CICSOPTS Assembly and IDMSINTC Link Edit (VSE/ESA)

```
// DLBL   idmslib,
// EXTENT ,nnnnnn
// LIBDEF *,SEARCH=(idmslib.sublib,cicslib.sublib)
// LIBDEF PHASE,CATALOG=idmslib.sublib
// OPTION CATAL
  PHASE   IDMSINTC,*
  INCLUDE DFHEAI
// EXEC ASMA90,SIZE=128K
CICSOPT macro statement
  END
/*
  INCLUDE  IDMSTRUE
  INCLUDE  DFHEAIO
  INCLUDE INTC6410
  ENTRY    STARTUP
// EXEC LNKEDT,SIZE=128K
/*
```

| Item | Description |
|---|---|
| cicslib.sublib | Name of the sublibrary within the library containing CICS modules |
| idmslib | Filename of the file containing CA-IDMS modules |
| idmslib.sublib | Name of the sublibrary within the library containing CA-IDMS modules |
| Nnnnnn | Volume serial identifier of appropriate disk volume |

**Note:** CICSOPT and the CICS macros and copy books must be accessible from the assigned source statement library.

Appendix

D

# New and Revised SQL Statements

## New and Revised SQL Statements

This appendix describes SQL statements and language elements that are new or that have changed in Release 15.0.

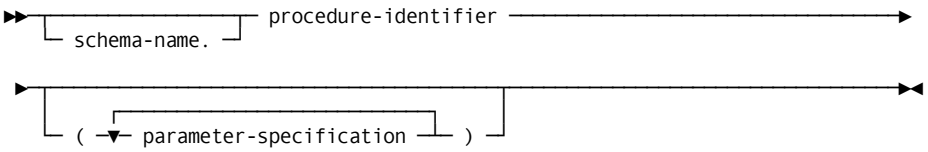
### Expansion of Procedure-Reference

#### Purpose

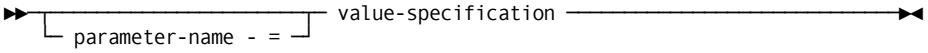
Represents a qualified or unqualified procedure identifier together with an optional set of parameter values.

If the procedure reference is contained in an SQL CALL or an SQL SELECT statement that is embedded in an application program, the procedure reference also identifies the target host variables into which the output parameter values will be returned.

#### Syntax




Expansion of parameter-specification



## Parameters

`schema-name`

Specifies the schema with which the procedure identified by procedure-identifier is associated.

See "Identifying entities in schemas" in the *SQL Reference Guide*, for information on using a schema name to qualify a procedure.

`procedure-identifier`

Identifies a procedure defined in the dictionary.

`parameter-specification`

Specifies a value to be assigned to a parameter of a procedure. If the procedure is contained in an SQL CALL or an SQL SELECT statement that is embedded in an application program and the value expression is a host variable, then the output value of the parameter will be returned into the specified host-variable.

Both the positional (with NO parameter-name) and the non-positional (with parameter-name) forms of parameter specification can be used in a single procedure reference.

If a non-positional parameter specification is used, all remaining parameter specifications in the parameter list MUST be non-positional. Positional parameter specifications are assumed to correspond to the declared parameters of a procedure in the sequence of their declaration.

`parameter-name`

Specifies the name of a parameter associated with the procedure.

`value-specification`

Any valid expression involving constants, host variables, database columns, and parameters of other procedures referenced in the same SQL statement.

## Usage

*Referencing procedures:* You can code references to SQL procedures in an SQL CALL statement.

During SQL CALL processing, CA-IDMS issues a call to the corresponding external routines. The output parameter values are returned as a result set.

A procedure can also be referenced in the FROM clause of a query-specification or SELECT statement, in the same manner as references to SQL tables, views, and table procedures.

If a procedure is referenced in a FROM clause, then the parameters of the procedure act as columns in an SQL table or view. You can reference them in SELECT list expressions and WHERE clauses. A procedure will return exactly one row of output or no output.

*Assigning parameter values with the WHERE clause:* An alternative method for assigning values to parameters of procedures is through the WHERE clause. An expression of the form parameter name = value specification coded in the WHERE clause is considered to be equivalent to a parameter assignment using procedure reference syntax. This allows procedure references to be coded without a parenthesized parameter list, just like standard table, view, or table procedure references.

This method is useful particularly if you are coding SQL statements in generic SQL environments, such as CA-Visual Express, which do not support CA-IDMS SQL extensions, such as procedures.

When you use the WHERE clause to assign parameter values, the following conditions must be met in order for the parameter to be assigned a value:

It must appear in an "=" comparison, not, for example, with >, <, >=, or <=.

The "=" comparison in which the parameter appears can be combined only with other factors in the WHERE clause using an AND operator. Use of an OR operator or preceding the "=" comparison with the NOT keyword will mean that no value is assigned to the parameter.

Refer to "Writing a procedure" for more information about assignment of values to procedure parameters.

## Examples

*Qualified procedure reference through the CALL statement:* In the following CALL statement, the procedure reference is qualified and one parameter value is supplied as a positional parameter for the first parameter of org():

```
call emp.get_bonus (127);
```

*Procedure reference with keyword parameter values:* In the following CALL statement, a value is supplied for the EMP_ID parameter using keyword notation:

```
call get_bonus (emp_id=127);
```

*Procedure reference through the SELECT statement:* In the following SELECT statement, a value is supplied for the first parameter associated with the GET_BONUS procedure:

```
select * from get_bonus (7);
```

Procedure reference through the SELECT statement with parameter values specified in the WHERE clause: In the following SELECT statement, parameter values are supplied through the WHERE clause. This example is identical to the example above that uses keyword notation.

```
select * from get_bonus
where emp_id=127
```

## ALTER PROCEDURE Statement

### Purpose

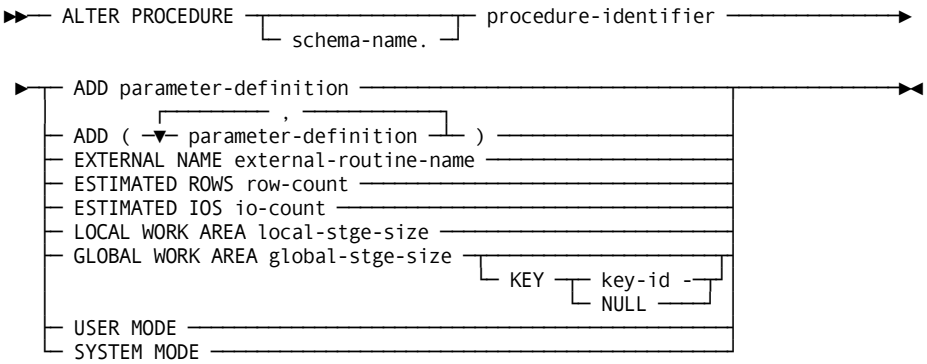
A data description statement that modifies the definition of a procedure in the dictionary. The ALTER PROCEDURE statement is a CA-IDMS extension of ANSI-standard SQL. Using the ALTER PROCEDURE statement, you can:

- Add a new parameter to a procedure
- Revise the estimated row and I/O counts
- Change the external name of the procedure
- Change the size and characteristics of the work areas passed to the procedure
- Change the execution mode of the procedure

### Authorization

To issue an ALTER PROCEDURE statement, you must either own or hold the ALTER privilege on the procedure named in the statement.

### Syntax



### Parameters

`procedure-identifier`

Specifies the name of the procedure being modified. Procedure-identifier must identify a procedure defined in the dictionary.

`schema-name`

Identifies the schema associated with the named procedure. If you do not specify a schema-name, it defaults to:

- The current schema associated with your SQL session, if the statement is entered through the Command Facility or executed dynamically
- The schema associated with the access module used at runtime, if the statement is embedded in an application program

`parameter-definition`

Defines one or more new parameters to be associated with the procedure. New parameters are added, in the order specified, after the last existing parameter.

Refer to "CREATE PROCEDURE," later in this chapter, for a description of parameter-definition.

`external-routine-name`

Specifies the one- to eight-character name of the program which CA-IDMS calls to process references to the procedure.

`row-count`

Specifies an integer value, in the range of 0 through 2,147,483,647, which represents the average number of rows that the procedure returns for a given set of input parameters.

`io-count`

Specifies an integer value, in the range of 0 through 2,147,483,647, which represents the average number of disk accesses that the procedure generates for a given set of input parameters.

`local-stge-size`

Specifies an integer, in the range of 0 through 32767, which represents the size, in bytes, of a local storage area which CA-IDMS allocates at runtime and passes to the procedure on each invocation.

CA-IDMS allocates a local storage area on the first call to a procedure for each SQL statement within a transaction or for a set of SQL statements related through reference to the same cursor. OPEN, CLOSE, and FETCH statements are related through a cursor. CA-IDMS passes the same local storage area to the procedure for all calls for one statement or related statements. CA-IDMS releases the local work area when the SQL statement has completed execution or at the time the cursor is closed.

`global-stge-size`

Specifies an integer, in the range of 0 through 32767, which represents the size, in bytes, of a global storage area which CA-IDMS allocates at runtime and passes to the procedure on each invocation.

A global storage area is allocated on the first call to a procedure within a transaction and is retained until the transaction terminates.

`key-id`

Specifies the one- to four-character identifier for the global storage area. CA-IDMS passes the same piece of global storage within a transaction to all procedures that have the same global storage key.

If you do not specify a storage key, its value remains unchanged. To remove a storage key, specify NULL as the key.

`USER MODE`

Specifies that the procedure should execute as a user-mode application program within CA-IDMS.

`SYSTEM MODE`

Specifies that the procedure should execute as a system-mode application program. To execute as a system mode application, the program must be:

- Written in Assembler
- Fully reentrant

## Examples

*Adding parameters to a procedure:* The ALTER PROCEDURE statement below adds two new parameters to the EMP.GET_BONUS procedure:

```
alter procedure emp.get_bonus
   add (start_month            char (2),
        start_year             char (2));
```

### For more information

See "CREATE PROCEDURE," later in this chapter, for more information about creating a procedure.

## CALL Statement

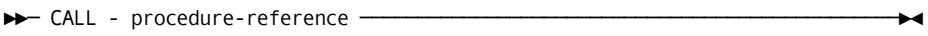
A data manipulation statement that executes a procedure. The values of output parameters are returned in the form of a one or more result rows. When the CALL statement is:

- Submitted through the command facility, the values in the result rows are displayed in tabular form
- Embedded in an application program, at most a single row can be returned and the value in the result row are stored in host variables
- Dynamically prepared, the result rows must be returned through a cursor just as if the prepared statement were a SELECT statement

### Authorization

To issue a CALL statement, you must either own or have the SELECT privilege on the procedure explicitly named in the statement.

### Syntax




### Parameters

`Procedure-reference`

Identifies the procedure that is to be invoked and the input values that are to be passed to the procedure.

### Usage

*Embedding a CALL statement:* When embedding a CALL statement in an application program, output values will be returned only for those parameters that are specified as host-variables. After executing the following statement, only the output value for the second parameter will be returned to the application program:

```
call myproc (5, :wk-out)
```

If the procedure updates the value of the first parameter and the application program needs to see that value, then both parameters must be specified as host-variables and the first host-variable should be set to 5 before executing the call statement:

```
move 5 to wk-val
call myproc (:wk-val, :wk-out)
```

*Initializing host variables:* It is important to initialize all host variables that are referenced in a CALL statement prior to its execution. Since all such host variables are treated as input values, failure to initialize such a host variable will result in a data exception if its value does not conform to its data type. If there is no value to be passed, the host-variable should be declared with a null indicator whose value is set to 01.

*Dynamically executing a CALL statement:* When describing the output from a dynamically prepared CALL statement, the SQLD field of the SQLDA will contain a count of the number of parameters defined for the procedure and the first SQLD entries within the SQLDA will contain a description of those parameters.

The output parameter values of a dynamically prepared CALL statement must be returned using a cursor. In other words, you must treat a dynamically prepared CALL statement as a dynamically prepared SELECT statement.

## CREATE PROCEDURE Statement

### Purpose

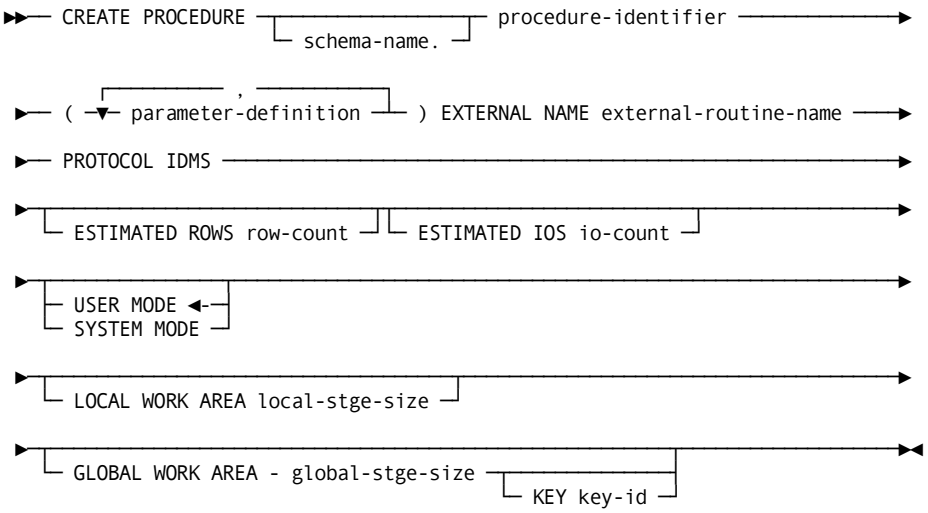
A data description statement that stores the definition of a procedure in the SQL catalog. You can then refer to the procedure in an SQL CALL statement or in an SQL SELECT statement just as you would a table procedure. These references result in CA-IDMS calls to the corresponding external routine. Although such routines can perform any action at all, you use them typically to manipulate data stored in some other organization (for example, in a non-SQL-defined database or in a set of VSAM files).

You use the formal parameters of a procedure definition like the columns of a table during a procedure invocation. You can input values in and return them from the procedure using column-like syntax.

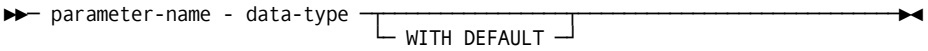
### Authorization

To issue a CREATE PROCEDURE statement, you must own the schema in which the procedure is being defined or hold the CREATE privilege on the named procedure.

## Syntax




**Expansion of** parameter-definition




## Parameters

`procedure-identifier`

Specifies the 1- to 18-character name of the procedure that you are creating. Procedure-identifier must:

- Be unique among the procedure, table, table procedure, and view identifiers within the schema associated with the procedure
- Follow conventions for SQL identifiers

`schema-name`

Specifies the schema name qualifier to be associated with the procedure. Schema-name must identify a schema defined in the dictionary. If you do not specify a schema-name, it defaults to:

- The current schema associated with your SQL session, if the statement is specified through the Command Facility or executed dynamically
- The schema associated with the access module used at runtime, if the statement is embedded in an application program

`parameter-definition`

Defines a parameter to be associated with the procedure. Parameters are passed to the procedure in the order in which they are specified. The list of parameters must be enclosed in parentheses. Multiple parameter definitions must be separated by commas.

Expanded syntax for parameter-definition is shown immediately following the CREATE PROCEDURE syntax.

`external-routine-name`

Specifies the one- to eight-character name of the program which will be called to process references to the procedure.

`PROTOCOL IDMS`

This is a mandatory parameter that specifies the protocol that will be used to invoke the procedure. The only protocol currently allowed is IDMS.

With the IDMS protocol the procedure is invoked with a single call for each reference to the procedure and the values that are returned, if any are treated as output values from a sub-program.

The PROTOCOL parameter is a CA-IDMS extension of ANSI-standard SQL.

`row-count`

Specifies an integer value, in the range 0 through 2,147,483,647, representing the average number of rows returned by the procedure for a given set of input parameters.

`io-count`

Specifies an integer value, in the range 0 through 2,147,483,647, representing the average number of rows returned by the procedure for a given set of input parameters.

`USER MODE`

Specifies that the procedure should execute as a user-mode application program within CA-IDMS. This is the default value unless SYSTEM MODE is specified.

`SYSTEM MODE`

Specifies that the procedure should execute as a system-mode application program. To execute as system mode, the program must be:

- Written in Assembler
- Fully reentrant

`local-stge-size`

Specifies an integer, in the range 0 through 32,767, which represents the size, in bytes, of a local storage area which will be allocated by CA-IDMS at runtime and passed to the procedure on each invocation.

CA-IDMS allocates a local storage area on the first call to a procedure for each SQL statement within a transaction or for a set of SQL statements which are related through reference to the same cursor (OPEN, FETCH, and CLOSE statements are related through a cursor). The same local storage area is passed to the procedure for all calls for one statement or related statements. When the SQL statement has completed execution or at the time at which the cursor is closed, the local work area is released.

**Note:** If you do not code a LOCAL WORK AREA clause, the default local storage size is 1024 bytes.

`global-stge-size`

Specifies an integer, in the range 0 through 32,767, which represents the size, in bytes, of a global storage area which will be allocated by CA-IDMS at runtime and passed to the procedure on each invocation.

A global storage area is allocated on the first call to a procedure within a transaction and is retained until the transaction terminates.

`key-ID`

Specifies the one- to four-character identifier for the global storage area. All procedures having the same global storage key are passed the same piece of global storage within a transaction.

If no storage key is specified, CA-IDMS allocates each procedure its own global storage area which will not be used for any other procedure within the transaction.

`parameter-name`

Specifies a 1- to 32-character name of a parameter to be passed to the procedure. Parameter-name must:

- Be unique within the procedure that you are defining
- Follow the conventions for SQL identifiers

All parameters are implicitly nullable. Input parameters can be assigned NULL as a parameter value and output parameters can return NULL.

`data-type`

Defines the data type for the named parameter. See "Expansion of data-type" in the *SQL Reference Guide* for expanded data-type syntax.

`WITH DEFAULT`

Directs CA-IDMS to pass a default value for the named parameter if no value for the parameter is specified.

The default value for a parameter is based on its data type:

| Column data type | Default value |
|---|---|
| CHARACTER | Blanks |
| VARCHAR | A character string literal with a length of zero (that is, ' ') |
| GRAPHIC | Double-byte blanks |
| VARGRAPHIC | A double-byte character string literal with a length of zero |
| DATE | The value in the CURRENT DATE special register |
| TIME | The value in the CURRENT TIME special register |
| TIMESTAMP | The value in the CURRENT TIMESTAMP special register |
| All numeric data types | 0 (zero) |

## Usage

*Influencing join strategies:* CA-IDMS uses estimated row and I/O counts in determining the cost of joining a procedure with other tables, views, procedures, or table procedures. To determine the optimal access strategy, CA-IDMS examines different sequences for retrieving information. By providing the estimated row and I/O counts for both the procedure and for each access key used by the procedure, CA-IDMS can select the optimal access strategy.

In determining the cost of a specific access strategy, CA-IDMS uses estimates provided in CREATE PROCEDURE unless input values are available for each of the parameters included in a key. If values are available for each of these parameters, CA-IDMS uses the estimates specified in the CREATE KEY statement instead of those specified in CREATE PROCEDURE.

See "CREATE KEY," in the *SQL Reference Guide*, for more information about CREATE KEY.

### Example

The CREATE PROCEDURE statement below defines a procedure.

```
create procedure emp.get_bonus
 ( emp_id                          unsigned numeric(4)    with default,
   bonus                             unsigned numeric(10)  with default,
   currency_bonus           char(3)                            with default )
external name getbonus
protocol idms;
```

### For more information

- On expanded procedure references, see "Expansion of procedure-reference" in the SQL Reference Guide.
- On coding the external routine, see Appendix F, "Defining and Using Procedures."

## DROP PROCEDURE Statement

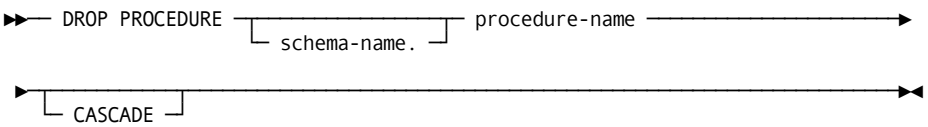
### Purpose

A data description statement that deletes the definition of the referenced procedure from the dictionary. The DROP PROCEDURE statement is a CA-IDMS extension of ANSI-standard SQL.

### Authorization

To issue a DROP PROCEDURE statement, you must own or have the DROP privilege on the procedure named in the statement.

### Syntax



### Parameters

`procedure-name`

Specifies the name of the procedure to be dropped. Procedure-name must identify a procedure defined in the dictionary.

`schema-name`

Identifies the schema associated with the specified procedure. If you do not specify a schema-name, the default value is:

- The current schema associated with your SQL session, if the statement is specified through the Command Facility or executed dynamically
- The schema associated with the access module used at runtime, if the statement is embedded in an application program

`CASCADE`

Directs CA-IDMS to delete any view definition that contains a reference to the procedure, either directly or nested within some other view reference.

### Example

The DROP PROCEDURE statement below removes the EMP.GET_BONUS procedure from the SQL catalog.

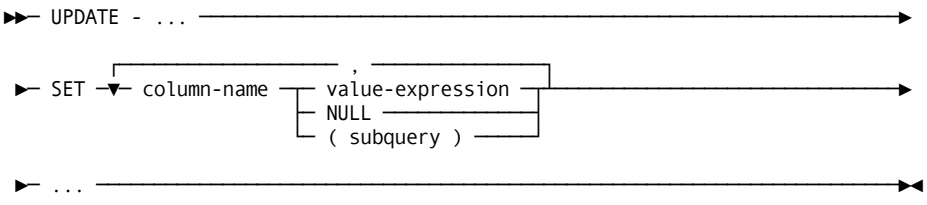
```
drop procedure emp.get_bonus cascade
```

### For more information

- On syntax for creating procedures, see "CREATE PROCEDURE" earlier in this chapter
- On defining and using procedures, see Appendix F, "Defining and Using Procedures"

## UPDATE Statement

### Syntax



### Parameters

`( subquery )`

Specifies the value to be stored in the named column. The subquery must return at most one row and the result table of the subquery must consist of a single column. See "expansion of subquery" in the SQL Reference for expanded **subquery** syntax.

### Usage

*Using a subquery as a source value*: If a subquery used as the value to be stored in a column returns no rows, the column will be set to the null value. If the column doesn't allow nulls, an exception will be raised.

Similarly, an exception will be raised if the subquery returns more than one row.

### Example

The following example sets the value of the SALARY_BUDGET column in the DEPARTMENT table based on the current salaries of all employees in the department.

```
update department d set salary_budget =
   (select 1.1 * sum (salary) from employee e
      where e.deptid = d.deptid)
```

Appendix

E

# Revised Dictionary Records

This appendix identifies the changes made to the dictionary records in support of Release 15.0 features. Each record that had changes is listed below followed by a table showing the fields that were added to the record.

## DMCL-1035

| Field | Picture | Description |
|---|---|---|
| 02 NUMJOURNALS-1035 | … | … |
| 02 SHAREDCACHE-1035 | X(16) DISPLAY | Default shared cache. |
| 02 LOCKENTRIES-1035 | S9(8) COMP SYNC | Number of entries in the coupling facility lock table. |
| 02 MEMBERS-1035 | X(8) BIT | Maximum number of members in the data sharing group. |
| 02 DATASHARE-1035 | X(1) DISPLAY | Data sharing indicator. A character 'Y' indicates that data sharing attributes have been specified for this DMCL. A character 'N' indicates that no data sharing attributes have been specified. |
| 02 FILLER | X(18) DISPLAY | |

# DMCLAREA-1036

| Field | Picture | Description |
|---|---|---|
| 02 PAGERESERVE-1036 | … | … |
| 02 DATASHARE-1036 | X(1) DISPLAY | Data sharing indicator. A character 'Y' indicates that this area is shared. A character 'N' indicates that it is not shared. A character 'D' indicates that its sharability status is determined by the data sharing indicator in the DMCLSEGMENT-1038 record. |
| 02 FILLER | X(39) DISPLAY | |

# DMCLSEGMENT-1038

| Field | Picture | Description |
|---|---|---|
| 02 WARMSTART-1038 | … | … |
| 02 DATASHARE-1038 | X(1) DISPLAY | Data sharing indicator. A character 'Y' indicates that areas in this segment are shared. A character 'N' indicates that they are not shared. |
| 02 SHAREDCACHE-1038 | X(16) DISPLAY | Shared cache. |
| 02 FILLER | X(23) DISPLAY | |

# LTRM-106

| Field | Picture | Description |
|---|---|---|
| 02 USER-COUNT-106 | … | … |
| 02 LTRM-FLAG3-106 | X(8) BIT | Flag byte 7.<br>X'80' — Suppress user newpage native<br>X'40' — Suppress user newpage nonnative<br>X'20' — Suppress SCS CR/LF at beginning or report<br>X'10' — Suppress newpage newline<br>X'08' — Auto newpage at end of native report<br>X'04' — Auto newpage at end of nonnative report<br>X'02' — Suppress newpage at beginning of native report<br>X'01' — Suppress newpage at beginning of nonnative report |
| 02 FILLER | X(5) DISPLAY | |

# LTRMLST-105

| Field | Picture | Description |
|---|---|---|
| 02 LTRM-FLAG2-105 | … | … |
| 02 LTRM-FLAG3-105 | X(8) BIT | Flag byte 7.<br>X′80′ — Suppress user newpage native<br>X′40′ — Suppress user newpage nonnative<br>X′20′ — Suppress SCS CR/LF at beginning of report<br>X′10′ — Suppress newpage newline<br>X′08′ — Auto newpage at end of native report<br>X′04′ — Auto newpage at end of nonnative report<br>X′02′ — Suppress newpage at beginning of native report<br>X′01′ — Suppress newpage at beginning of nonnative report |
| 02 FILLER | X(2) DISPLAY | |

# PROG-051

The only change to the PROG-051 record is the addition of a new value for the PROG-FLAG4-051 field. A value of X′08′ indicates that the program is checked out.

# QUEUE-DCQ-138

| Field | Picture | Description |
|---|---|---|
| 02 QUEUE-FLAG-138 | … | … |
| 02 QUEUE-FLAG2-138 | X(8) BIT | Flag byte |
| 02 QUEUE-DC-SYS-NAME-138 | X(8) DISPLAY | Name of system owning a local queue. |
| 02 FILLER | X(14) DISPLAY | |

# SYS-041

| Field | Picture | Description |
|---|---|---|
| 02 DEADLOCK-STALL-INT-041 | … | … |
| 02 RECOVERY-WAIT-041 | S9(4) COMP SYNC | Recovery wait time in wall clock seconds |
| … | … | … |
| 02 SYS-FLAG6-041 | … | |

| Field | Picture | Description |
|---|---|---|
| 02 SYS-FLAG7-041 | X(8) BIT | Flag byte 7.<br>X'80' — Suppress user newpage native<br>X'40' — Suppress user newpage nonnative<br>X'20' — Suppress SCS CR/LF at beginning of report<br>X'10' — Suppress newpage newline<br>X'08' — Auto newpage at end of native report<br>X'04' — Auto newpage at end of nonnative report<br>X'02' — Suppress newpage at beginning of native report<br>X'01' — Suppress newpage at beginning of nonnative report |
| 02 SYS-FLAG8-041 | X(8) BIT | Flag byte 8 |
| … | … | … |
| 02 DEDICATED-ERUS-041 | … | |
| 02 QUIESCE-WAIT-041 | S9(4) COMP SYNC | Quiesce wait time in wall clock seconds |
| 02 FILLER | X(44) DISPLAY | |

# SYSMO-170

| Field | Picture | Description |
|---|---|---|
| 02 DEADLOCK-STALL-INT-170 | … | … |
| 02 RECOVERY-WAIT-170 | S9(4) COMP SYNC | Recovery wait time in wall clock seconds |
| … | … | … |
| 02 SYS-FLAG6-170 | … | |
| 02 SYS-FLAG7-170 | X(8) BIT | Flag byte 7.<br>X'80' — Suppress user newpage native<br>X'40' — Suppress user newpage nonnative<br>X'20' — Suppress SCS CR/LF at beginning of report<br>X'10' — Suppress newpage newline<br>X'08' — Auto newpage at end of native report<br>X'04' — Auto newpage at end of nonnative report<br>X'02' — Suppress newpage at beginning of native report<br>X'01' — Suppress newpage at beginning of nonnative report |
| 02 SYS-FLAG8-170 | X(8) BIT | Flag byte 8. |
| … | … | … |
| 02 DEDICATED-ERUS-170 | … | |
| 02 QUIESCE-WAIT-170 | S9(4) COMP SYNC | Quiesce wait time in wall clock seconds |
| 02 FILLER | X(44) DISPLAY | |

# TASK-025

| Field | Picture | Description |
|---|---|---|
| 02 AREA-ACQ-RETRY-025 | … | … |
| 02 EXT-WAIT-025 | S9(4) COMP SYNC | External wait time in wall clock seconds |
| 02 QUIESCE-WAIT-025 | S9(4) COMP SYNC | Quiesce wait time in wall clock seconds |
| 02 FILLER | X(30) DISPLAY | |

# TASKLST-023

| Field | Picture | Description |
|---|---|---|
| 02 AREA-ACQ-RETRY-023 | … | … |
| 02 EXT-WAIT-023 | S9(4) COMP SYNC | External wait time in wall clock seconds |
| 02 QUIESCE-WAIT-023 | S9(4) COMP SYNC | Quiesce wait time in wall clock seconds |
| 02 FILLER | X(34) DISPLAY | |

Appendix

F

# Defining and Using Procedures

## When to use a procedure

Procedure support provides the ability to define and call procedures using a simpler interface than that of table procedures. This support is a subset of the SQL standard specification for external procedures and provides a way of invoking routines using a remote procedure call paradigm.

To use this feature, the following steps must be taken:

- Define the procedure using the new CREATE PROCEDURE statement.
- Write the procedure in COBOL, PLI, or Assembler following the guidelines outlined below. It may also be possible to use an existing program as a procedure.
- If necessary, define the program to a CA-IDMS system.
- Invoke the procedure with an SQL CALL statement or from within a query-specification or a SELECT statement.

## Defining a procedure

You define a procedure using the CREATE PROCEDURE statement. In the example below, the procedure GET_BONUS is named and associated with schema EMP. The name of the program to be called to service a CALL request of the procedure, CALCSALR, is specified in the EXTERNAL NAME parameter. The PROTOCOL IDMS specifies that the procedure is defined and called using the IDMS protocol. The parameters to be passed to and from the procedure are listed. Each parameter definition consists of a name and a data type.

```
CREATE PROCEDURE EMP.GET_BONUS
( EMP_ID                          USIGNED NUMERIC (4),
  START_DATE                DATE,
  SALARY                          UNSIGNED NUMERIC (9))
EXTERNAL NAME CALCSAL
PROTOCOL IDMS;
```

See "CREATE PROCEDURE" in Appendix D for syntax and parameters used in defining procedures. See Appendix G, "Sample COBOL Procedure," for a more detailed example of using CREATE PROCEDURE.

# Accessing a procedure

You access table procedures using an SQL CALL statement or using a query-specification or a SELECT statement. During SQL CALL processing, CA-IDMS issues a call to the corresponding external routines. The output parameter values are returned as a result set.

A procedure can also be referenced in the FROM clause of a query-specification or SELECT statement, in the same manner as references to SQL tables, views, and table procedures.

If a procedure is referenced in a FROM clause, then the parameters of the procedure act as columns in an SQL table or view. You can reference them in SELECT list expressions and WHERE clauses. A procedure will return exactly one row of output or no output. You can reference table procedures any place in which a table reference is permitted.

Access to procedure is controlled in the same way as for a table procedure. GRANT and REVOKE statements on a resource type of TABLE are used to give and remove SELECT or DEFINE privileges on a procedure.

# Procedure parameters

## Parameters in procedure references of the SQL CALL statement

The recommended and easiest way of specifying input parameter values is within the procedure reference itself in the SQL CALL statement. When invoked through the Command Facility, the CALL statement will result in a set of output values for each parameter defined to the procedure. In embedded SQL a CALL statement will result in an output value for each parameter specified as a host variable.

You specify parameter values supplied on a procedure reference either positionally or as keyword/value pairs. You can also combine them with WHERE clause references to form the set of values that will be passed to the procedure.

Examples of specifying parameter values: The example below shows different ways in which you can specify input parameter values using the SQL CALL statement.

```
CALL EMP.GET_BONUS (EMP_ID = 127, START_YEAR= '1998')
CALL EMP.GET_BONUS (127, '1998')
```

## Parameters in procedure references in query-specification and SELECT statements

The parameters associated with a procedure are treated like columns of a table. You can specify them within the column list of a SELECT or a query-specification or the search criteria of a WHERE clause. Additionally, you can specify parameter values within the procedure reference itself.

**Column List References:**

| Parameters referenced in | Specify |
|---|---|
| Column list of a SELECT statement | The columns that will be returned to the invoking application |

## WHERE clause references

WHERE clause references to parameters are used to filter the output of the procedure. Each time a procedure returns a set of output values, they are evaluated against the selection criteria specified in the WHERE clause and a non-conforming "row" will result in an SQLSTATE of No Data for the initiating SQL request.

Additionally, WHERE clause parameter references are used to pass input values to the procedure. If you specify selection criteria in the form of an "=" comparison (that is, parameter = value), value is passed to the procedure. Other types of selection criteria such as IN predicates or comparison predicates with > or < operators have no effect on the value of the parameters passed to the procedure.

Specifically, a reference to a parameter in a WHERE clause results in an input value being passed to the procedure only if:

- It appears within an equality test
- The equality test is not combined with other predicates in the WHERE clause through the use of the OR operator
- The equality test is not preceded by the NOT operator

*WHERE clause parameter references:* The examples below illustrate how a parameter reference in a WHERE clause affects the value passed to the procedure:

| WHERE clause | Parameter value P1 | P2 |
|---|---|---|
| P1 = 1 | 1 | -null- |
| P1 < 1 | -null- | -null- |
| P1 = C1 | C1 | -null- |
| P1 = 2 AND P2 = 3 | 2 | 3 |
| P1 = 2 AND P2 > 3 | 2 | -null- |
| P1 = 2 OR P2 = 3 | -null- | -null- |
| P1 IN (2, 3, 8) | -null- | -null- |

*Difference between procedure reference and WHERE clause:* One difference exists between parameter values specified through a WHERE clause and those specified within the procedure reference. Parameter values specified within the procedure reference are not used to filter the output from the procedure as is the case for those specified within the WHERE clause. Parameter values specified within the procedure reference affect only the input to the procedure and not the output from the procedure. Therefore, the above three select statements are equivalent only if the procedure enforces the conditions specified through the procedure reference.

**Note:** The sample procedure in Appendix G, "Sample COBOL Procedure," does not enforce any criteria other than those that it uses to navigate the database.

# Writing a procedure

The program associated with a procedure can be written in COBOL, PL/I, or Assembler. When called, the program is passed a fixed parameter list consisting of the parameters specified on the procedure definition as well as additional parameters used for communication between CA-IDMS and the procedure.

Whenever a reference to a procedure is made, CA-IDMS calls the program associated with the procedure to service the request. The procedure responds by processing the input parameters. An error condition can optionally be set in SQLSTATE.

CA-IDMS performs transaction and session management automatically in response to requests that the originating application issues. Changes to the database made by a procedure are committed or rolled out together with other changes made within the SQL transaction. No special action is required of the procedure in order to ensure that this occurs.

The section below discusses writing a procedure in detail.

See Appendix G, "Sample COBOL Procedure," for an example of a table procedure written in COBOL.

## Calling arguments

The following sets of arguments are passed when a procedure is called:

- One argument for each of the parameters specified on the procedure definition, passed in the order in which the parameters were declared
- One argument for each null indicator associated with a parameter specified in the procedure definition, passed in the order in which the parameters were declared
- A set of common arguments used for communications between CA-IDMS and the procedure

The first two sets of arguments will vary from one procedure to another. They are used to pass selection criteria and insert/update values to the procedure and result values from the procedure.

The last set of arguments, shown in the table below, is the same for all procedures.

| Argument | Contents |
|---|---|
| Result Indicator (fullword) | Not used |
| SQLSTATE (CHAR (5)) | Status code returned by the procedure:<br>The initial values is always 00000<br>00000 — Indicates success<br>01Hxx — Indicates a warning<br>02000 — Indicates no more rows<br>38xxx — Indicates an error |
| Procedure Name (CHAR (18)) | Name of the procedure |
| Explicit Name | Not used |
| Message Text (CHAR (80)) | Message text returned by the procedure and displayed by CA-IDMS in the event of an error or warning |
| SQL Command Code (fullword) | Always 16, indicating a Fetch SQL request |
| SQL Operation Code (fullword) | Always 16, indicating a "next row" request |
| Instance Identifier (fullword) | Not meaningful for procedures |

| Argument | Contents |
|---|---|
| Local Work Area (User-defined) | A user-defined working storage area |
| Global Work Area (user-defined) | A user-defined storage area shared by one or more procedures or table procedures |

## Parameter arguments

On entry to the procedure, the value of the arguments corresponding to the parameters defined on the CREATE PROCEDURE statement are as follows:

- Non-null parameters contain one of the following:
  - The parameter values specified on the procedure reference
  - The selection criteria specified in the WHERE clause
  - The datatype-specific default value if WITH DEFAULT was specified in the procedure definition
- All other parameters contain nulls (that is, the null indicator for the parameter is negative).

On exit, the procedure is expected either to have set the value of the parameter arguments and their indicators appropriately or to have set an SQLSTATE value indicating no-more-rows. If an indicator parameter is set to -1, CA-IDMS ignores the value of the corresponding parameter.

## Local work area

Another parameter passed on each call to a procedure is a local work area.

CA-IDMS allocates the local work area just before calling the procedure and frees it immediately after the procedure exits. When the local work area is allocated, it is initialized to binary zeros.

## Global work area

A global work area is a storage area which is shared across one or more procedures or table procedures within a transaction. Each global work area has an associated key which is either:

- The four-character identifier specified on the GLOBAL WORK AREA clause
- The fully-qualified name of the procedure if no identifier was specified

All procedures executing within a transaction and having the same global storage key share the same global work area.

All procedures within an invoking SQL transaction should update the database through only one run unit or SQL transaction to avoid deadlocking. Typically, an update procedure will use a global work area in order to share the subschema control or SQL session identifier across procedures. A retrieval-only procedure might instead use only a local work area opening the run unit or SQL session and terminating it on exit.

## Special considerations for procedures

The special considerations of table procedures apply also to procedures. See the *SQL Reference Guide* for more information.

Appendix

G

# Sample COBOL Procedure

## About this appendix

This appendix provides:

- A sample procedure definition.
- A sample procedure written in COBOL. This program is included on the CA-IDMS installation tape. It requires the employee demo database and assumes use of a VS COBOL II compiler.
- Samples of procedure invocation.

## Sample procedure definition

The example below shows a table procedure definition.

```
create procedure demoemp1.get_bonus
  ( emp_id                              unsigned numeric(4)  with default,
    bonus                               unsigned numeric(10) with default,
         currency_bonus char(3)                              with default)
    external name getbonus
    protocol idms;
```

## Sample procedure

The sample program shown below is included on the CA-IDMS installation tape. This program requires the SQL employee demo database.

```
*COBOL PGM SOURCE FOR GETBONUS
*RETRIEVAL
*DMLIST
 IDENTIFICATION DIVISION.
 PROGRAM-ID.                          GETBONUS.
 AUTHOR.                                   DEFJE01.
 INSTALLATION.                        SYSTEM71.
 DATE-WRITTEN.                        06/25/99.
*-----------------------------------------------------------------------*
*                                                                       *
```

```
* GETBONUS will return the sum of all bonus amounts for a             *
* given employee.                                                     *
*   Parameters:                                                       *
* EMP_ID          : input parameter must contain employee id          *
* BONUS           : output parameter returns sum of bonus             *
* CURRENCY-BONUS  : output parameter returns currency symbol          *
*                   or 'ERR' in case of an error condition            *
* These parameters are assumed to have been defined                   *
* 'WITH DEFAULT' in the procedure definition, so that null            *
* indicators do not need to be defined and processed                  *
*---------------------------------------------------------------------*
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
*SOURCE-COMPUTER.                    IBM WITH DEBUGGING MODE.
*

 DATA DIVISION.
*
 WORKING-STORAGE SECTION.
  01 ERROR-STATUS                                          PIC X(4).
*---------------------------------------------------------------------*
*                                                                     *
*---------------------------------------------------------------------*
 LINKAGE SECTION.
* Procedure parameters
  EXEC SQL BEGIN DECLARE SECTION END-EXEC.
  77 EMP-ID                                                 PIC 9(4).
  77 BONUS                                                  PIC 9(10).
  EXEC SQL END DECLARE SECTION END-EXEC.
  77 CURRENCY-BONUS                                         PIC X(3).
* Other parameters do not need to be specified
*---------------------------------------------------------------------*
 PROCEDURE DIVISION USING EMP-ID, BONUS, CURRENCY-BONUS.
 0000-MAINLINE.
 MOVE '$' TO CURRENCY-BONUS.

 EXEC SQL
   SELECT SUM(BONUS_AMOUNT) INTO :BONUS
     FROM DEMOEMPL.BENEFITS
     WHERE EMP_ID = :EMP-ID
 END-EXEC

 IF SQLSTATE NOT = '00000'
   MOVE 'ERR' TO CURRENCY-BONUS.

 EXIT PROGRAM.
 STOP RUN.
```

# Samples of procedure invocation

The first four examples are all equivalent. The last example returns an error indication.

```
call demoemp1.get_bonus(1234);
EMP_ID               BONUS  CURRENCY_BONUS
   1234                    6530    $
```

1 row processed

```
call demoemp1.get_bonus(emp_id = 1234);
```

```
EMP_ID                 BONUS   CURRENCY_BONUS
  1234                      6530   $
```

1 row processed

```
select * from demoemp1.get_bonus where emp_id = 1234;
EMP_ID                  BONUS    CURRENCY_BONUS
  1234                       6530      $
```

1 row processed

```
select * from demoemp1.get_bonus(emp_id = 1234);
EMP_ID                     BONUS    CURRENCY_BONUS
  1234                          6530    $
```

1 row processed

```
call demoemp1.get_bonus(0);
EMP_ID                      BONUS    CURRENCY_BONUS
         0                          0          ERR
```

1 row processed

Appendix

# H Revised System Tables

This appendix identifies the changes made to the system tables in support of Release 15.0 features. Each table that had changes is listed below followed by a description of the columns that were added.

## SYSTEM.DMCL

| Column name | Description | Data Type | Null Specification |
|---|---|---|---|
| NUMJOURNALS | ... | … | … |
| SHAREDCACHE | Default shared cache. | CHARACTER(16) | NOT NULL |
| LOCKENTRIES | Number of entries in the coupling facility lock table. | INTEGER | NOT NULL |
| MEMBERS | Maximum number of members in the data sharing group. | BINARY(1) | NOT NULL |
| DATASHARE | Data sharing indicator.<br>■ 'Y' — data sharing attributes have been specified.<br>■ 'N' — data sharing attributes have not been specified. | CHARACTER(1) | NOT NULL |
| ONCONNECTLOSS | Connection loss indicator.<br>■ 'A' — Abend<br>■ 'N' — Noabend | CHARACTER(1) | NOT NULL |
| FILLER | Reserved for future use. | CHARACTER(17) | NOT NULL |

# SYSTEM.DMCLAREA

| Column name | Description | Data Type | Null Specification |
|---|---|---|---|
| PAGERESERVE | … | … | … |
| DATASHARE | Data sharing indicator.<br>■ 'Y' — the area is eligible to be shared for update<br>■ 'N' — the area is not shared<br>■ 'D' — the area's sharability is determined by that of its segment | CHARACTER(1) | NOT NULL |
| FILLER | Reserved for future use. | CHARACTER(39) | NOT NULL |

# SYSTEM.DMCLSEGMENT

| Column name | Description | Data Type | Null Specification |
|---|---|---|---|
| WARMSTART | … | … | … |
| DATASHARE | Data sharing indicator.<br>■ 'Y' — all areas in the segment are eligible to be shared for update<br>■ 'N' — no areas in the segment are shared | CHARACTER(1) | NOT NULL |
| SHAREDCACHE | Default shared cache. | CHARACTER(16) | NOT NULL |
| FILLER | Reserved for future use. | CHARACTER(23) | NOT NULL |